Uniform Convergence of the Deep Galerkin Method for the Mean Field Control Problem

Jake Hofgard

A thesis presented for the degree of

Bachelor of Science with Honors

Department of Mathematics

Stanford University

May 2024

Acknowledgements

I would like to thank Prof. Asaf Cohen, my advisor through the University of Michigan Mathematics REU, for providing feedback, advice, and support over the course of the last year as this project has developed from its conceptualization to a nearly complete publication. Additionally, I would like to thank Prof. Lenya Ryzhik, my honors thesis advisor at Stanford University, for his advice and feedback. I am also grateful for the support and encouragement provided by my peers in the math department, including (but certainly not limited to) Shaunak Bhandarkar, Owen Brass, Jasper Shogren–Knaak, Kai Fronsdal, Aditi Talati, and Sidhart Krishnan. Finally, I would like to thank my partner, Natasha Johnson, and my family for their continued support over the course of my time at Stanford.

Abstract

We establish the convergence of the deep Galerkin method (DGM), a deep learning-based numerical method for solving high-dimensional nonlinear PDEs, for Hamilton–Jacobi–Bellman (HJB) equations that arise from the study of mean field control problems (MFCPs). Based on a recent characterization of the value function of the MFCP as the unique viscosity solution of an HJB equation on the simplex, we establish both an existence and convergence result for the DGM. First, we show that the loss functional of the DGM can be made arbitrarily small given that the value function of the MFCP possesses sufficient regularity. Then, we show that if the loss functional of the DGM converges to zero, the corresponding neural network approximators must converge uniformly to the true value function on the simplex. We also provide numerical experiments demonstrating the DGM's ability to generalize to high-dimensional HJB equations.

Contents

Chapter 1

Introduction

We consider the convergence of a deep learning-based numerical method for solving Hamilton–Jacobi–Bellman (HJB) equations that arise from the study of mean field control problems (MFCPs). The specific class of HJB equations that we consider are first-order, nonlinear PDEs on the $(d-1)$–dimensional simplex S_d of the following form:

$$\begin{aligned} &-\partial_t V(t,m) + \sum_{i \in [\![d]\!]} m_i H^i(t,m,D^i V(t,m)) = 0, \\ &V(T,m) = \sum_{i \in [\![d]\!]} m_i g^i(m). \end{aligned} \tag{1.1}$$

Above, $[\![d]\!] := \{1, \ldots, d\}$, $V : [0,T] \times S_d \to \mathbb{R}$ is the *value function* (i.e., the optimal cost) of an associated MFCP, H^i is a Hamiltonian, and D^i denotes a derivative along the simplex, namely $D^i V(t,m) = (\partial_{m_j - m_i} V(t,m))_{j \in [\![d]\!]}$. Note that on the simplex, directional derivatives are only permitted in the directions $e_j - e_i$, where e_i denotes the ith standard basis vector in $\mathbb{R}^d$. We focus on solving Equation (1.1) numerically using neural networks.

Our numerical approach is primarily based on the deep Galerkin method (DGM), as first introduced by [1]. In particular, we solve the above class of PDEs using a deep learning-based algorithm motivated by classical finite element methods. However, instead of finding a basis of functions to approximate solutions as in the classical method, we utilize a sufficiently rich class of neural networks to approximate solutions to Equation (1.1). In addition to proving the convergence of our numerical scheme under the assumptions outlined in Section 3.3, we provide numerical experiments that demonstrate the validity of this approach. First, however, we motivate the study of MFCPs by introducing the corresponding N-agent optimization problem in Section 1.1. In turn, we introduce the MFCP model in Section 1.2, describe the current field of solving high-dimensional PDE using deep learning in Section 1.3, and outline our main results in Section 1.4. Section 1.5 describes common notation used throughout this thesis, and Section 1.6 provides a roadmap for the remainder of this thesis.

1.1 N-Agent Optimization Problems

In the N-agent problem, we assume that the dynamics in

$$\mathbb{P}(X_{t+h}^k = j \mid \mathbf{X}_t = \mathbf{x}) = Q_{x_k,j}(t, \beta_k(t, \mathbf{x}), \mu_{\mathbf{x}}^N)h + o(h) \tag{1.2}$$

hold, and N cooperating agents aim to minimize the common cost

$$J^N(\boldsymbol{\beta}) = \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\left[\int_0^T f(t, X_t^k, \beta_k(t, X_t^k), \mu_t^N)dt + g(X_k^T, \mu_T^N)\right], \tag{1.3}$$

where the running cost f and terminal cost g depend on the agents' empirical distribution, with coordinates given by

$$\mu_{i,t}^N = \frac{1}{N}\sum_{k=1}^{N} \mathbb{1}_{X_t^k = i}$$

for $i \in \{1, \ldots, d\}$. Let $\mathcal{A}$ denote the set of admissible controls, which we take to be measurable, Markovian feedback controls. Note that in the N-agent setting, a control $\boldsymbol{\beta} \in \mathcal{A}^N$ consists of a sequence of N controls, one for each agent. Next, we define the associated value function (which we aim to describe as the solution to an appropriate HJB equation) by

$$v^N(t, \mathbf{x}) = \inf_{\boldsymbol{\beta} \in \mathcal{A}^N} \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\left[\int_t^T f(s, X_s^k, \beta_k(s, X_s^k), \mu_s^N)ds + g(X_k^T, \mu_T^N) \,\middle|\, \mathbf{X}_t = \mathbf{x}\right] \tag{1.4}$$

$$=: \inf_{\boldsymbol{\beta} \in \mathcal{A}^N} J^N(t, \boldsymbol{\beta}, \mathbf{x}), \tag{1.5}$$

assuming that minimizing the cost in (1.3) is the goal of the agents. We make the same assumptions as [2], discussed in detail below, so that the value function v belongs to

$\mathcal{C}^{1,1}([0,T] \times S_d)$; see [2, Theorem 3.5]. In particular, this ensures that the HJB equation for the N-agent optimization problem (and the MFCP) has a classical solution.

Below, let $[\![d]\!]^N := \{1, \dots, d\}^N$. Then, given $\mathbf{x} \in [\![d]\!]^N$, we define $[\mathbf{x}^{-k}, j] \in [\![d]\!]^N$ for $j \in [\![d]\!]^N$ by

$$[\mathbf{x}^{-k}, j]_\ell = \begin{cases} x_\ell & \ell \neq k, \\ j & \ell = k. \end{cases}$$

In turn, given $u : [\![d]\!]^N \to \mathbb{R}^d$, we define $\Delta^k u \in \mathbb{R}^d$ by $\Delta^k u(\mathbf{x})_j = u([\mathbf{x}^{-k}, j]) - u(\mathbf{x})$.

We utilize the notation $Q_{i,\bullet}$ to denote the ith row of the transition rate matrix $(Q_{i,j})_{i,j \in [\![d]\!]^N}$. From this, we define the pre-Hamiltonian $\mathcal{H}^i : [0,T] \times \mathcal{A} \times S_d \times \mathbb{R}^d \to \mathbb{R}$ for each $i \in [\![d]\!]$ by

$$\mathcal{H}^i(t, a, m, z) := -\langle Q_{i,\bullet}(t, a, m), z \rangle - f(t, i, a, m). \tag{1.6}$$

In turn, the corresponding Hamiltonian is given by

$$H^i(t, m, z) = \sup_{a \in \mathcal{A}} \mathcal{H}^i(t, a, m, z) \tag{1.7}$$

for each $i \in [\![d]\!]$. With this notation out of the way, we can state the HJB equation for the N-agent optimization problem.

Proposition 1.1.1. *The value function v^N defined in Equation (1.4) is $\mathcal{C}^1$ in time and uniquely solves the HJB equation*

$$\begin{aligned} -\frac{\partial v^N}{\partial t}(t, \mathbf{x}) + \frac{1}{N} \sum_{k=1}^N H^{x_k}(t, \mu^N_{\mathbf{x}}, N \Delta^k v^N(t, \mathbf{x})) = 0, \\ v^N(T, x) = \frac{1}{N} \sum_{k=1}^N g(x_k, \mu^N_{\mathbf{x}}). \end{aligned} \tag{1.8}$$

In [2], it is shown that the above N-agent optimization problem is in fact equivalent to a single optimization problem in terms of the empirical distribution of the agents. This problem is given by a time-inhomogeneous Markov chain with dynamics given by

$$\mathbb{P}\left(\mu^N_{t+h} = m + \frac{1}{N}(\delta_j - \delta_i) \,\middle|\, \mu^N_t = m \right) = N m_i Q_{i,j}(t, \alpha_N(t, i, m), m) h + o(h), \tag{1.9}$$

where $m \in S_d^N$, $i \neq j \in [\![d]\!]$, and each $\alpha^N(t, \cdot, m) \in \mathcal{A}$ is now a *single* control that depends only on the state of each agent and the empirical distribution of the agents. The cost functional the problem is now given by the expression

$$J^N(\alpha_N, t, m) = \mathbb{E}\left[\int_t^T \sum_{i \in [\![d]\!]} m_{i,t}\mu_{i,s}^N f(s, i, \alpha_N(s, i, \mu_s^N), \mu_s^N) ds + \sum_{i \in [\![d]\!]} \mu_{i,T}^N g^i(\mu_T^N)\right]. \quad (1.10)$$

Defining

$$D_j^{N,i} v(m) := N\left(v(m + \frac{1}{N}(\delta_j - \delta_i)) - v(m)\right),$$

we have the following analogous proposition.

Proposition 1.1.2. *The value function V^N for the control problem described by Equations* (1.9) – (1.10) *is $\mathcal{C}^1$ in time and uniquely solves the HJB equation*

$$\begin{aligned} -\frac{\partial V^N}{\partial t}(t, m) + \sum_{i \in [\![d]\!]} m_i H^i(t, m, D^{N,i} V^N(t, m)) = 0, \\ V^N(T, m) = \sum_{i \in [\![d]\!]} m_i g^i(m). \end{aligned} \quad (1.11)$$

As noted above and shown in [2, Proposition 2.6], the HJB equations in Proposition 1.1.1 and Proposition 1.1.2 are equivalent. The two problems are equivalent in the sense that, using optimal controls from either the original N-agent optimization problem or the reformulated problem in which the agents only interact through their empirical distribution, the agents will follow the same trajectories.

1.2 Mean Field Control Problems

MFCPs are limiting models for cooperative games with a finite, but large, number of interacting agents attempting to minimize a common cost, as outlined in the preceding section. Namely, the study of MFCPs is motivated by situations in which a large number of agents aim to achieve a common goal. Applications arise in the control of autonomous vehicles and drones [3, 4], efficient real-time streaming between devices [5], decentralized and centralized crowd motion [6, 7], analysis of wireless local-area networks (WLANs) [8, 9], among many other related applications. The inception of limiting models for many-agent games can be traced back to Aumann's work in 1964 and Schmeidler's influential contribution in 1973; see [10] and [11]. In the context of stochastic dynamical games, mean field game (MFG) theory, which now encapsulates both MFCPs and their counterpart mean field games (MFGs), was first developed by Lasry and Lions in [12] and independently by Huang, Caines, and Malhamé in [13].

At this point, we distinguish between MFCPs and MFGs. While both are frameworks for approximating games involving numerous players, the former models cooperative games whereas the latter models non-cooperative games. MFCPs represent optimal control problems for McKean–Vlasov dynamics, where the value function (optimal cost) is often described by a HJB equation. Conversely, in MFGs, Nash equilibria are approximated through a fixed point of a best response map, and the value function (which describes the cost under equilibrium for the MFG) is characterized by the so-called "master equation." For more details on the master equation, including its formulation, relevance to MFGs, and deep learning-based methods for obtaining solutions, see [14].

Further work by Carmona and Delarue [15] and Andersson and Djehiche [16] provided a probabilistic framework for mean field games through the analysis of forward-backward stochastic differential equations (FBSDEs) of a McKean–Vlasov type. Recent work by Pham and Wei also derived a dynamic programming principle and HJB equation for the MFCP by framing it as a deterministic control problem of a Fokker–Planck equation; see [17, 18]. Even more recently, work by Cecchin [2] established important existence and uniqueness results for the HJB equation corresponding to the finite-state MFCP that we consider here. For a comprehensive overview of MFG theory, MFCPs, and their respective relationships to stochastic games and control problems, see [19, 20, 21].

Given current advances in deep learning, there has been recent progress towards solving both MFGs and MFCPs numerically using deep learning [22, 23]. Building on the theoretical results in [2], we aim to provide a novel pathway towards solving MFCPs using deep learning. In practice, finding the optimal control in N-agent stochastic control problems quickly becomes intractable as N grows. As a result, the MFCP is of theoretical and practical interest in stochastic control. Here, we provide a short motivation for its structure. As N approaches infinity, the N-agent optimization problem resembles an optimization problem involving a single *representative* agent whose dynamics follow a controlled continuous-time Markov chain of the form:

$$\mathbb{P}(X_{s+h} = j \mid X_s = i) = \alpha_{i,j}(s)h + o(h) \qquad \text{as} \qquad h \to 0^+, \tag{1.12}$$

where the feedback control of the representative agent $\alpha_{i,j}(s)$ is a measurable, Markovian function, standing for the transition rate of the above process from state i to j, both in $[\![d]\!]$. Throughout, we consider only rate matrices satisfying $\alpha_{i,j}(s) \in [0, M]$ for all $i \neq j, i, j \in [\![d]\!]$, $s \in [t_0, T]$ and some constant $M > 0$. We refer to this set of rate matrices as *admissible controls*. We denote the set of admissible controls by $\mathcal{A} := [0, M]^{d^2}$. Above, α_i refers to the ith row of the transition matrix α. At this point, there may appear to be a gap between the richness of the feedback controls allowed in the N-agent problem versus the single feedback control of the MFCP. However, this is not the case. Given the symmetry among the N agents in the original optimization problem, one can consider substituting their transition rates with functions of the form $\bar{\beta}^k(s, X_s^k, \mu_s^N; \cdot)$ that depend on the empirical distribution of agents. Guided by the propagation of chaos result in [2], we anticipate the convergence of $(\mu_t^N)_{t\in[t_0,T]}$ toward a deterministic flow of measures, approximating the distributions of each coordinate in $\mathbf{X}$. This deterministic framework implies that, in the limit, we can anticipate the control's dependence on both the current time t and the present state i.

With this aside out of the way, note that as in the N-agent case, the representative agent aims to minimize the cost

$$J(t_0, m_{t_0}, \alpha) := \mathbb{E}\left[\int_{t_0}^{T} f(s, X_s, \alpha_{X_s}(s), \mathrm{Law}(X_s))ds + g(X_T, \mathrm{Law}(X_T))\right], \tag{1.13}$$

where m_{t_0} is the initial distribution of the process X. Above, $\mathrm{Law}(X_{t_0}) = m_{t_0} \in S_d$ is predetermined. This problem can be alternatively described as a deterministic control problem in terms of the Fokker–Planck equation for $(\mu_s := \mathrm{Law}(X_s))_{s\in[t_0,T]}$. In particular, for a given admissible control α, the measure μ uniquely solves the ordinary differential equation (ODE):

$$\begin{aligned} &\frac{d}{dt}\mu_t^i = \sum_{j\in[\![d]\!]} \left(\mu_t^j \alpha_{j,i}(s) - \mu_t^i \alpha_{i,j}(s)\right), \qquad i \in [\![d]\!], \quad t \in [t_0, T], \\ &\mu_{t_0} = m_{t_0}, \end{aligned} \tag{1.14}$$

and the cost functional for the deterministic control problem becomes

$$J(t_0, m_{t_0}, \alpha) := \int_{t_0}^{T} \sum_{i \in [\![d]\!]} f(s, i, \alpha_i(s), \mu_s)\mu_s^i ds + \sum_{i \in [\![d]\!]} g^i(\mu_T)\mu_T^i. \tag{1.15}$$

Given an initial time $t_0 \in [0, T]$, the *value function* $V : [0, T] \times S_d \to \mathbb{R}$ of the deterministic control problem is then defined by

$$V(t, m) = \inf_{\alpha \in \mathcal{A}} J(t, \alpha, m). \tag{1.16}$$

By standard optimal control arguments via an appropriate dynamic programming principle, presented in Section 2.3 below, this gives rise to the HJB equation in Equation (1.1). As in the N-agent optimization problem, we utilize the notation $Q_{i,\bullet}$ to denote the ith row of the transition rate matrix $(Q_{i,j})_{i,j \in [\![d]\!]^N}$. From this, we define the pre-Hamiltonian $\mathcal{H}^i : [0, T] \times \mathcal{A} \times S_d \times \mathbb{R}^d \to \mathbb{R}$ for each $i \in [\![d]\!]$ by

$$\mathcal{H}^i(t, a, m, z) := -\langle Q_{i,\bullet}(t, a, m), z \rangle - f(t, i, a, m). \tag{1.17}$$

In turn, the corresponding Hamiltonian is given by

$$H^i(t, m, z) := \sup_{a \in \mathcal{A}} \mathcal{H}^i(t, a, m, z) \tag{1.18}$$

for each $i \in [\![d]\!]$. We also define a map $H : [0, T] \times S_d \times \mathbb{R}^d \to \mathbb{R}$ by

$$H(t, m, z) := \sum_{i \in [\![d]\!]} m_i H^i(t, m, z), \tag{1.19}$$

Note that we refer to both H^i and H as Hamiltonians. To avoid confusion, we occasionally employ the term PDE-Hamiltonian specifically for H. Under the assumptions presented in Section 2.1 below, the above equation has a unique maximizer [24], which we denote by $a^*(t, i, m, z)$ for each $i \in [\![d]\!]$. In turn, the *unique* optimal control for the MFCP is given by

$$\alpha_{i,j}^*(t, m) = a_j^*(t, i, m, D^i V(t, m)), \tag{1.20}$$

where the directional derivative of the value function $D^i V(t, m)$ on the simplex S^d is as defined in Equation (1.1). Additionally, the dynamics of the process μ under the optimal control satisfy the dynamics in Equation (1.14) upon replacing the control therein with the optimal control from Equation (1.20), as shown in [2].

Importantly, recent work described in full detail in Section 2 rigorously establishes the connection between this N-agent stochastic control problem and the corresponding MFCP. In particular, Cecchin [2] formulates the MFCP in terms of the deterministic optimal control of a Fokker–Planck equation as in Equations (1.14) and (1.15). The resulting deterministic control problem yields a dynamic programming principle and the HJB equation (1.1), solved by the value function of the MFCP. Above, however, we obtained two HJB equations: one for the N-agent optimization problem and one for the MFCP. Both HJB equations have unique viscosity solutions, as shown in [2]. In turn, [2] establishes an explicit rate of convergence between V^N, the value function for the N-agent problem, and V, the value function for the MFCP. In particular, a convergence rate of $1/\sqrt{N}$ is established.

In a similar vein, Kolokoltsov [25] obtains a convergence rate of $1/N$ under additional regularity assumptions. Notably, [2] places no regularity assumptions on the value function V (other than Lipschitz continuity) and, in the most general case, does not require convexity of the running or terminal costs of the MFCP. Given the relationship between the N-agent problem and the MFCP, by approximately solving the HJB equation for the MFCP, one could also use our method to approximately solve the N-agent problem.

1.3 Numerical Solutions to High-Dimensional PDE

We aim to construct a numerical scheme for efficiently solving the HJB equation associated with the MFCP. In particular, as the dimension d increases, the so-called "curse of dimensionality" prevents standard numerical schemes (e.g., Monte Carlo methods, mesh-based algorithms, etc.) from solving the HJB equation in a tractable manner. However, recent advancements in deep learning present promising options for solving high-dimensional, nonlinear PDEs such as the HJB equation in Equation (1.1).

Two leading methods have been presented for solving parabolic PDEs that resemble the HJB equation: the deep Galerkin method (DGM) and deep backward stochastic differential equations (BSDE). The primary focus of this thesis is the DGM. This approach, first introduced in [1], models itself after classical finite element methods for solving low-dimensional PDEs. However, the DGM is a mesh-free method; instead of creating basis functions that approximate the solution to a PDE from a mesh, the DGM utilizes neural network approximations that only depend on the parameters, architecture, and activation function between layers of the network. The loss functional of the DGM with L^2-loss attempts to minimize the L^2-error of both the PDE and the terminal condition during training, ultimately learning the parameters that best approximate the solution of the PDE. In [1], the authors introduce the DGM, illustrate its ability to numerically solve high-dimensional, nonlinear PDEs, and provide a convergence guarantee for second-order, nonlinear parabolic PDEs.

The second popular deep learning-based method for solving high-dimensional PDEs, introduced in [26] and expanded upon in [27, 28], exploits the connection between nonlinear parabolic PDEs and backward SDEs that can be exploited via a so-called nonlinear Feynman–Kac formula. In turn, the resulting backward SDE can be solved numerically by recursively using a sequence of neural networks to solve the SDE along a specified discretization of the time interval in question, starting with the terminal condition of the original PDE. Although this method, often referred to as deep BSDE, is likely applicable to our context, we defer further consideration of deep backward schemes to future work.

Since 2020, several review articles have brought up the potential for solving MFCPs with deep learning methods, including the DGM and deep BSDE, but they note that so far that no convergence guarantee has been provided for these methods; see [29, 30]. Similarly, [31] provides a broad sampling of applications of deep BSDE-like algorithms to common problems in stochastic control, again without providing a convergence guarantee for their proposed method. For MFGs, the picture is more clear. For instance, [14] analyzes the convergence of both a DGM-type algorithm and a BSDE-type algorithm for the master equation, which characterizes the value function for the MFG equilibrium. Interestingly, the authors of [14] propose a *deterministic* deep backward scheme in place of the more standard stochastic setting first introduced in [26]. The deep backward master equation (DBME) algorithm in [14] instead relies on a key consistency relation between the solution to the master equation and the measure that solves a Fokker–Planck equation for the MFG system. By performing a time discretization of the Fokker–Planck equation and solving it on each resulting interval using a neural network, the DBME algorithm can efficiently leverage the consistency relation from [14, Proposition 1] to solve the master equation. Although we have access to a Fokker–Planck equation in the MFCP, as described in Equation (1.14), the relationship between the value function that solves the HJB equation in Equation (1.1) and the measure that solves the forward equation in Equation (1.14) is not as clear as in the case of MFGs. Thus, establishing the convergence of a DBME-type algorithm for MFCPs remains a topic of future work.

1.4 Main Results

We show that the convergence guarantee provided in [1] extends to the (first-order) HJB equation (1.1), relying on the theory of viscosity solutions for HJB equations to obtain the desired convergence. Importantly, our class of HJB equations does not fall into the class of second-order nonlinear parabolic PDEs considered in [1], so we provide a different proof technique. In order to leverage the theory of viscosity solutions in our proof technique, we modify the loss functional used in [1] to approximate the L^{∞}-norm rather than the L^2-norm. Viscosity solutions are often referred to as L^{∞}-weak solutions, as they respect convergence with respect to the uniform norm [32]. The work by [1] establishes the uniform convergence of neural network approximators to the actual solution a family of partial differential equation. However, this result is obtained by introducing additional assumptions of uniform boundedness and equicontinuity on the neural networks, which we can also bypass using L^{∞}-loss. With this reformulation of the DGM loss, we first prove an existence result by showing in Theorem 3.2.4 that a sequence of neural networks taking the loss functional of the DGM to zero exists. With Theorem 3.3.2, we then prove that for such a sequence of networks, the neural network approximators converge uniformly to the true value function for the MFCP on $[0, T] \times S_d$.

To obtain this result, we utilize a standard argument from the theory of viscosity solutions, relying on an appropriate definition of viscosity solutions to Equation (1.1) on the simplex and a corresponding comparison principle. By establishing that a sequence of neural network approximators are viscosity solutions to a sequence of perturbed HJB equations and applying the comparison principle to two suitable upper and lower limits of the neural network approximators, we can conclude that the sequence of neural network approximators taking the DGM loss functional to zero must converge uniformly to the classical solution to Equation (1.1). The main steps in our proof are outlined as follows:

(1) First, in Theorem 3.2.4, we show that there exists a neural network φ that makes the DGM loss (defined in Section 3.1) arbitrarily small.

(2) From this, in Corollary 3.2.5, we establish that there exists a sequence of neural network approximators $\{\varphi^n\}_{n\in\mathbb{N}}$ such that $\varphi^n \to V$ uniformly on $[0,T] \times S_d$ as $n \to \infty$, where V is the unique classical solution to Equation (1.1).

(3) Then, we handle the technical details of working with a PDE on the $(d-1)$–dimensional simplex S_d. In particular, the interior of S_d is empty, so we instead work on $\widehat{S}_d$, the projection of the simplex onto $\mathbb{R}^{d-1}$. These technical details are discussed in Proposition 3.3.1, and we work on $\widehat{S}_d$ for the remainder of the proof in order to fully leverage the power of viscosity solutions. Throughout, we use *hat* notation to denote mathematical objects that are defined on $\widehat{S}_d$ rather than on S_d. For example, $\widehat{V}$ is the version of V, defined on $\widehat{S}_d$.

(4) Finally, we arrive at our main result in Theorem 3.3.2, which establishes that if $\{\widehat{\varphi}^n\}_{n\in\mathbb{N}}$ is a sequence of neural network approximators taking the DGM loss to zero (which we know exists from Corollary 3.2.5), then $\widehat{\varphi}^n \to \widehat{V}$ uniformly on $[0,T] \times \widehat{S}_d$.

(5) The proof of Theorem 3.3.2 relies on a comparison principle for viscosity solutions, as presented in [33, Theorem 3.3]. In particular, we construct a viscosity subsolution $\overline{V}$ and a viscosity supersolution $\underline{V}$ on $[0,T] \times \widehat{S}_d$ that satisfy $\underline{V} \leq \overline{V}$ by construction, and $\overline{V} \leq \underline{V}$ by the comparison principle. By the uniqueness result for viscosity solutions to the HJB equation [2, Theorem 9], this implies that $\overline{V} = \underline{V} = \widehat{V}$, and the construction of $\overline{V}$ and $\underline{V}$ provides uniform convergence as a byproduct.

We note that although our proof technique is only applied to PDEs of the form presented in Equation (1.1), the general approach is likely applicable to a broader class of HJB equations that admit viscosity solutions.

1.5 Notation

As noted above, we define $[\![d]\!] := \{1, \ldots, d\}$. The $(d-1)$–dimensional simplex S_d is given by

$$S_d := \left\{ (m_1, \ldots, m_d) \in \mathbb{R}^d : m_i \geq 0, \quad \sum_{i=1}^{d} m_i = 1 \right\}. \tag{1.21}$$

On the simplex S_d, we define directional derivatives for a function $\varphi : S_d \to \mathbb{R}$ by

$$D^i f(m) := (\partial_{m_j - m_i} f(m))_{j \in [\![d]\!]},$$

where derivatives are in the direction of $e_j - e_i$ for the corresponding standard basis vectors of $\mathbb{R}^d$.

Unless otherwise specified, Ω will refer to a subset of $\mathbb{R}^n$ (measurable, but not necessarily open). As usual, $L^p(\Omega)$ consists of the measurable functions on Ω such that

$$\|f\|_p := \left(\int_\Omega |f(x)|^p dx \right)^{1/p} < \infty,$$

where the integral above is taken with respect to the Lebesgue measure on $\mathbb{R}^d$, restricted to Ω. For $p = \infty$, we instead take $L^\infty(\Omega)$ to be the space of measurable functions on Ω whose essential supremum is bounded. We say that a function is differentiable on Ω if it is differentiable on the interior of Ω and extends to a differentiable function on an open neighborhood of $\mathbb{R}^n$ that contains Ω. As is standard, $\mathcal{C}(\Omega)$ refers to the space of continuous functions on Ω, equipped with the uniform norm:

$$\|f\|_\infty := \sup_{x \in \Omega} |f(x)|.$$

Below, when we refer to *uniform convergence*, we always refer to convergence in the above norm.

We use the notation $\mathcal{C}^{1,1}(\Omega)$ to denote (continuously-differentiable) functions that have Lipschitz-continuous first derivatives on Ω. See [2] for a more thorough discussion of this class of functions as they relate to the HJB equation in Equation (1.1). Other spaces of functions that we refer to throughout include the space of continuously-differentiable functions with k continuously-differentiable derivatives, denoted by $\mathcal{C}^k(\Omega)$. Finally, we define the standard norm on $\mathcal{C}^k(\Omega)$ by

$$\|f\|_{\mathcal{C}^k(\Omega)} := \max_{|\alpha| \leq m} \sup_{x \in \Omega} |D^\alpha f(x)|.$$

1.6 Organization

The remainder of this thesis is structured as follows. Section 2 introduces the relevant assumptions placed upon the MFCP that ensure sufficient regularity of the value function. In particular, Section 2.1 presents all assumptions that we place on the MFCP, Section 2.2 discusses known existence, uniqueness, and convergence results for the MFCP, and Section 2.3 contains detailed derivations of the HJB equations for the N-agent problem and the MFCP and N-agent problem. Section 3 presents our modified version of the DGM algorithm for the HJB equation, the resulting convergence guarantees, and all corresponding proofs. Within, Section 3.1 presents the DGM algorithm along with several technical details surrounding its implementation and efficiency, Section 3.2 provides necessary background on the universal approximation power of neural networks, Section 3.3 contains our primary convergence results and the associated proofs, and Section 3.4 compares our results to previously established convergence guarantees for the DGM. Finally, Section 4 presents selected numerical results from a JAX implementation of the DGM in Section 4.1 before concluding with a brief exploration of algorithmic modifications that may improve the efficiency of the DGM in Section 4.2.

Chapter 2

Assumptions and Known Results for the MFCP

2.1 Assumptions for the MFCP

We start by assuming the following, drawing upon the notation introduced in the preceding section.

Assumption A. *The function* $F : [0,T] \times [0,M]^{d^2} \times S_d \to \mathbb{R}$*, given by*

$$F(t, a^1, \ldots, a^d, m) := \sum_{i \in [\![d]\!]} m_i f(t, i, a^i, m)$$

is continuous and there exists a constant $C > 0$*, such that for any* $t, s \in [0,T]$*,* $a \in [0,M]^d$*, and* $m, p \in S^d$*,*

$$|F(t,a,m) - F(s,a,p)| \leq C(|t-s| + |m-p|).$$

Assumption B. *For each* $i \in [\![d]\!]$*, the running cost* f *is continuously differentiable in* a*,* $\nabla_a f$ *is Lipschitz continuous with respect to* m*, and there exists* $\lambda > 0$ *such that*

$$f(t,i,b,m) \geq f(t,i,a,m) + \langle \nabla_a f(t,i,a,m), b-a \rangle + \lambda |b-a|^2.$$

Assumption C. *We have that:*

(C1) Defining $G : S_d \to \mathbb{R}$ by

$$G(m) := \sum_{i \in [\![d]\!]} m_i g^i(m),$$

we have that $F(\cdot, a, \cdot) \in \mathcal{C}^{1,1}([0,T] \times S_d)$ and $G \in \mathcal{C}^{1,1}(S_d)$.

(C2) The function

$$[0,T] \times [0,\infty)^{d\times d} \times Int(S_d) \ni (t,w,m) \mapsto \sum_{i \in [\![d]\!]} m_i f\left(t, i, \left(\frac{w_{i,j}}{m_i}\right)_{j \neq i}, m\right)$$

is convex in (w, m).

(C3) G is convex in m.

Note that under Assumption (C), G is differentiable and hence Lipschitz continuous on compact sets: for any $m, p \in S_d$,

$$|G(m) - G(p)| \leq C|m - p|$$

for some $C > 0$, which we can take to be the same constant as in Assumption (A). We provide an example of an MFCP in Section 4 for which the running and terminal costs satisfy all three of the above assumptions.

2.2 Existence, Uniqueness, and Convergence Results for the MFCP

Under the above assumptions, Cecchin [2] derived a series of useful results for the MFCP. We begin with a uniqueness result for solutions to the above HJB equations, from [2, Theorem 2.9].

Proposition 2.2.1. *Let V be the value function for the deterministic control problem in* (1.15). *Then, if:*

(1) Assumption (A) holds, V is the unique viscosity solution of Equation (1.1) *on S_d and V is Lipschitz continuous in (t, m).*

(2) Assumptions (A) and (B) hold, there exists an optimal control to the deterministic MFCP.

(3) Assumptions (A) – (C) hold, then $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$ is the unique classical solution of the HJB equation in Equation (1.1).

Although the above theorem is the most important result from [2] for our work, Cecchin also presents several convergence results connecting the MFCP to the N-agent optimization problem. In particular, we have, from [2, Theorem 2.10] the following result concerning convergence of the value function V^N for the N-agent optimization problem in (1.2) to the value function V for the MFCP. Note that this result only requires the standard stochastic control assumptions from Assumption (A).

Proposition 2.2.2. *Under Assumption (A), we have that*

$$\max_{(t,m)\in[0,T]\times S_d} \left|V^N(t,m) - V(t,m)\right| \le \frac{C}{\sqrt{N}}$$

for all $N \in \mathbb{N}$.

Next, [2, Theorem 2.11] contains a similar convergence result for the cost obtained by the optimal control for the MFCP.

Proposition 2.2.3. *Let $\varepsilon > 0$ and $N \in \mathbb{N}$. Then, under Assumption (A), if $\alpha : [0,T] \to \mathcal{A}$ is an ε-optimal control for the MFCP,*

$$J^N(\alpha) \le \inf_{\alpha^N \in \mathcal{A}^N} J^N(\alpha^N) + \frac{C}{\sqrt{N}} + \varepsilon,$$

where J^N is the cost functional for the N-agent optimization problem in (1.3).

Finally, Cecchin also presents a propagation of chaos of result, describing the connection between the optimal trajectory of the N-agent optimization problem and the MFCP in [2, Theorem 2.13].

Proposition 2.2.4. *If Assumption (B) holds and $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$, then*

$$\mathbb{E}\left[\sup_{t\in[0,T]} |\mu_t^N - \mu_t|\right] \le \frac{C}{N^{1/9}},$$

where μ^N *is the process in* (1.9) *and* μ *is the optimal trajectory of the MFCP, satisfying* (1.14).

At this point, we note that while Propositions 2.2.2, 2.2.3, and 2.2.4 are not directly relevant to our main results in Section 3, they demonstrate that solving the MFCP *can* reliably approximate solutions to the N-agent control problem, with quantitative estimates for the rate of convergence between the two. Consequently, given an N-agent optimization problem as in Section 1.1, one can translate the problem into an MFCP, obtain an accurate solution for Equation (1.1) using the DGM, and by Proposition 2.2.2, estimate the approximation error of the value function for the N-agent problem, which solves Equation 1.8.

From there, a standard optimal control argument, as outlined in [34], will recover a sequence of optimal controls for the N-agent optimization problem from the value function for Equation 1.8, thereby solving the N-agent optimization problem. Critically, when solving Equation (1.1) using the DGM, our convergence results ensure that the DGM will produce a reliable approximation, with respect to the uniform norm on $[0, T] \times S_d$, to the classical solution to Equation (1.1), assuming that such a solution exists.

2.3 Derivations of HJB Equations

In this section, we present the derivations of the HJB equations for both the N-agent optimization problem and the MFCP, first introduced in Section 1. We begin with the HJB equation in Proposition 1.1.1, corresponding to the standard N-agent optimization problem.

Proof of Proposition 1.1.1. Recall the definition of the generator of a continuous-time Markov chain: if $\varphi : \{1, \dots, d\}^N \to \mathbb{R}$ and $T_{t,h}\varphi(\mathbf{x}) := \mathbb{E}[\varphi(\mathbf{X}_{t+h}) \mid \mathbf{X}_t = \mathbf{x}]$ denotes the Feller semigroup of the process, then we define the infinitesimal generator of the process by

$$\mathcal{L}_t^{N,\boldsymbol{\beta}}\varphi(\mathbf{x}) := \lim_{h \downarrow 0} \frac{T_{t,h}\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{h}. \tag{2.1}$$

We can now explicitly compute the limit in (2.1) given the dynamics in (1.2). In particular, we observe that

$$T_{t+h}\varphi(\mathbf{x}) = \mathbb{E}[\varphi(\mathbf{X}_h) \mid \mathbf{X}_t = \mathbf{x}] = \sum_{k=1}^{N}\sum_{j\neq x_k}\left[Q_{x_k,j}(h,\beta_k(h,\mathbf{x}),\mu_{\mathbf{x}}^N)h + o(h)\right]\varphi([\mathbf{x}^{-k},j]).$$

Assuming that $\varphi : \{1,\dots,d\}^N \to \mathbb{R}$ is bounded, the fact that $o(h)/h \to 0$ as $h \downarrow 0$ now implies that

$$\begin{aligned}\mathcal{L}_t^{N,\boldsymbol{\beta}}\varphi(\mathbf{x}) = \lim_{h\downarrow 0}\frac{T_{t,h}\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{h} &= \sum_{k=1}^{N}\sum_{j\neq x_k} Q_{x_k,j}(t,\beta_k(t,\mathbf{x}),\mu_{\mathbf{x}}^N)\left[\varphi([\mathbf{x}^{-k},j]) - \varphi(\mathbf{x})\right]\\ &= \sum_{k=1}^{N}\langle Q_{x_k,\bullet}(t,\beta_k(t,\mathbf{x},\mu_{\mathbf{x}}^N)),\Delta^k\varphi(\mathbf{x})\rangle\\ &= \frac{1}{N}\sum_{k=1}^{N}\langle Q_{x_k,\bullet}(t,\beta_k(t,\mathbf{x},\mu_{\mathbf{x}}^N)), N\Delta^k\varphi(\mathbf{x})\rangle.\end{aligned}$$

From Assumption (A), we have that the transition rate $Q_{i,j}$ is continuous on $[0,T]\times A\times S_d$, allowing us to pass the limit inside the transition rate above.

The next step is the key step that allows us to apply Itô's formula, or rather, Dynkin's formula [34]. In particular, recall that if $(X_t)_{t\in[0,T]}$ is a Feller process (as is the case here) and $\mathcal{L}_0^{N,\boldsymbol{\beta}}\varphi$ is well-defined, then the process given by

$$M_t^{\varphi} := \varphi(X_t) - \varphi(X_0) - \int_0^t \mathcal{L}_0^{N,\boldsymbol{\beta}}\varphi(X_s)ds.$$

is a martingale adapted to the canonical filtration of the process $(X_t)_{t\in[0,T]}$; see [35, Chapter 17], for instance. Now, if v^N denotes the value function of the N-agent stochastic control problem, we can now apply Dynkin's formula (noting that $(M_t^{v^N})_{t\in[0,T]}$ is now a martingale, even if $(X_t)_{t\in[0,T]}$ may not be) to see that

$$v^N(t+h,\mathbf{X}_{t+h}) = v^N(t,\mathbf{X}_t) + \int_t^{t+h}\left(\frac{\partial v^N}{\partial t} + \mathcal{L}_t^{N,\boldsymbol{\beta}}v^N\right)(s,\mathbf{X}_s)ds.$$

Taking conditional expectations, we find that

$$\mathbb{E}[v^N(t+h,\mathbf{X}_{t+h}) \mid \mathbf{X}_t = \mathbf{x}] = v^N(t,\mathbf{x}) + \mathbb{E}\left[\int_t^{t+h}\left(\frac{\partial v^N}{\partial t} + \mathcal{L}_t^{N,\boldsymbol{\beta}}v^N\right)(s,\mathbf{X}_s)ds \,\middle|\, \mathbf{X}_t = \mathbf{x}\right]. \tag{2.2}$$

Here, we remark that the above expression, obtained via Itô's formula, may also contain a local martingale that disappears upon taking expectations.

Now, for any control $\boldsymbol{\beta}'$, we observe that for all $0 \le t < r \le T$,

$$\begin{aligned} v^N(t,\mathbf{x}) &= \inf_{\boldsymbol{\beta}\in A^N} J^N(t,\boldsymbol{\beta},\mathbf{x}) \\ &\le J^N(t,\boldsymbol{\beta}',\mathbf{x}) \\ &= \frac{1}{N}\sum_{k=1}^N \mathbb{E}\left[\int_t^T f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N)ds + g(X_k^T,\mu_T^N) \,\middle|\, \mathbf{X}_t = \mathbf{x}\right] \\ &= \frac{1}{N}\sum_{k=1}^N \mathbb{E}\left[\int_r^T f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N)ds + g(X_k^T,\mu_T^N) \,\middle|\, \mathbf{X}_t = \mathbf{x}\right] \\ &\quad + \frac{1}{N}\sum_{k=1}^N \mathbb{E}\left[\int_t^r f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N)ds \,\middle|\, \mathbf{X}_t = \mathbf{x}\right] \\ &= \mathbb{E}[J^N(r,\boldsymbol{\beta}',\mathbf{X}_r) \mid \mathbf{X}_t = \mathbf{x}] \\ &\quad + \frac{1}{N}\sum_{k=1}^N \mathbb{E}\left[\int_t^r f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N)ds \,\middle|\, \mathbf{X}_t = \mathbf{x}\right] \end{aligned}$$

by the Markov property and the tower property. Now, assume that the control $\boldsymbol{\beta}'$ is after time r so that $J^N(r,\boldsymbol{\beta}',\mathbf{X}_r) = v^N(r,\mathbf{X}_r)$. We then obtain, with $r = t+h$,

$$\begin{aligned} v^N(t,\mathbf{x}) &\ge \mathbb{E}[v^N(t+h,\mathbf{X}_{t+h}) \mid \mathbf{X}_t = \mathbf{x}] \\ &\quad + \frac{1}{N}\sum_{k=1}^N \mathbb{E}\left[\int_t^{t+h} f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N)ds \,\middle|\, \mathbf{X}_t = \mathbf{x}\right]. \end{aligned} \tag{2.3}$$

Applying the law of iterated expectation and plugging this inequality into the inequality in (2.2), we see that

$$\mathbb{E}\left[\int_t^{t+h} \frac{1}{N}\sum_{k=1}^N f(s,X_s^k,\beta_k'(s,X_s^k),\mu_s^N) + \left(\frac{\partial v^N}{\partial t} + \mathcal{L}_t^{N,\boldsymbol{\beta}} v^N\right)(s,\mathbf{X}_s)ds\right] \ge 0.$$

Dividing by h and taking the limit as $h \downarrow 0$ as in [34], we obtain via the mean value theorem that

$$\begin{aligned}0 &\geq -\frac{\partial v^N}{\partial t}(t,\mathbf{x}) + \left(-\mathcal{L}_t^{N,\boldsymbol{\beta}} v^N(t,\mathbf{x}) - \frac{1}{N}\sum_{k=1}^N f(t, x_k, \beta_k(t,\mathbf{x}), \mu_t^N)\right) \\ &= -\frac{\partial v^N}{\partial t}(t,\mathbf{x}) + \frac{1}{N}\left(-\sum_{k=1}^N \langle Q_{x_k,\bullet}(t, \beta_k(t,\mathbf{x},\mu_{\mathbf{x}}^N)), N\Delta^k v^N(t,\mathbf{x})\rangle - \sum_{k=1}^N f(t,x_k,\beta_k(t,\mathbf{x}),\mu_t^N)\right),\end{aligned}$$

with equality when the optimal control $\boldsymbol{\beta}^\star$ is chosen. As a result, we have that

$$\begin{aligned}0 = &-\frac{\partial v^N}{\partial t}(t,\mathbf{x}) \\ &+ \sup_{\boldsymbol{\beta}\in\mathcal{A}^N}\left(-\sum_{k=1}^N \langle Q_{x_k,\bullet}(t, \beta_k(t,\mathbf{x},\mu_{\mathbf{x}}^N)), N\Delta^k v^N(t,\mathbf{x})\rangle - \sum_{k=1}^N f(t,x_k,\beta_k(t,\mathbf{x}),\mu_t^N)\right).\end{aligned}$$

Thus, by the preceding definition of the Hamiltonian, we therefore conclude that the value function v^N satisfies

$$-\frac{\partial v^N}{\partial t}(t,\mathbf{x}) + \frac{1}{N}\sum_{k=1}^N H^{x_k}(t, \mu_{\mathbf{x}}^N, N\Delta^k v^N(t,\mathbf{x})) = 0.$$

Additionally, because the terminal cost is given by $\frac{1}{N}\sum_{k=1}^N g(\mathbf{X}_k^T, \mu_T^N)$, the associated terminal condition for the above system of ODEs must be

$$v^N(T,x) = \frac{1}{N}\sum_{k=1}^N g(x_k, \mu_{\mathbf{x}}^N).$$

The above derivation yields the HJB equation for the N-agent optimization problem. Uniqueness and the stated regularity then follow from [2, Proposition 2.3]. □

Next, we derive the HJB equation for reformulation of the N-agent problem, presented in Proposition 1.1.2.

Proof of Proposition 1.1.2. The generator of this Markov chain is instead given by

$$\mathcal{L}_t^{N,\alpha_N} v(m) = N\sum_{i,j\in[\![d]\!]} m_i Q_{i,j}(t, \alpha_N(t,i,m), m)\left[v(m + \frac{1}{N}(\delta_j - \delta_i) - v(m)\right], \qquad (2.4)$$

but the HJB equation for the new value function $V^N(t,m)$ can be derived precisely as above. Consequently, the same reasoning as in the proof of Proposition 1.1.1 (albeit with a different generator) yields the HJB equation

$$\begin{aligned}0 &= -\frac{\partial V^N}{\partial t}(t,m) + \sup_{\alpha_N \in \mathcal{A}} \left(-\mathcal{L}_t^{N,\alpha_N} V^N(t,m) - \sum_{i \in [\![d]\!]} m_i f(t,i,a^i,m) \right) \\ &= -\frac{\partial V^N}{\partial t}(t,m) + \sum_{i \in [\![d]\!]} m_i H^i(t,m,D^{N,i}V^N(t,m))\end{aligned}$$

in terms of the Hamiltonian defined in [2]. Finally, we have the following terminal condition, coming from the terminal cost term in (1.10):

$$V^N(T,m) = \sum_{i \in [\![d]\!]} m_i g^i(m).$$

Again, uniqueness and the stated regularity are shown in [2, Proposition 2.6]. □

Finally, it remains to derive the HJB equation for the MFCP itself, as stated in Proposition 2.2.1.

Proof. In Section 1, we reduced the MFCP to a deterministic control problem. Thus, we may apply the dynamic programming principle (DPP) as usual. Below, we also use the fact that, by assumption, $\mathcal{A} = [0,M]^{d^2}$. Now, if the value function of the MFCP problem is given by

$$V(t,\mu) = \inf_{\alpha \in \mathcal{A}} J(t,\alpha,\mu),$$

then the DPP (as stated and derived for the deterministic control problem in [36]) states that

$$V(t,\mu_t) = \inf_{\alpha \in \mathcal{A}} \left(\int_t^{t+h} \sum_{i \in [\![d]\!]} f(s,i,\alpha^i(s),\mu_s)\mu_s^i ds + V(t+h,\mu_{t+h}) \right).$$

Above, the control α is such that $\alpha : [0,T] \to \mathcal{A}$. Subtracting $V(t,\mu_t)$ from both sides and dividing by h, we obtain

$$0 = \inf_{\alpha \in \mathcal{A}} \left(\frac{1}{h} \int_t^{t+h} \sum_{i \in [\![d]\!]} f(s,i,\alpha^i(s),\mu_s)\mu_s^i ds + \frac{V(t+h,\mu_{t+h}) - V(t,\mu_t)}{h} \right)$$

Taking the limit as $h \downarrow 0$ then yields

$$\begin{aligned}
0 &= \inf_{a\in[0,M]^d}\left(\sum_{i\in[\![d]\!]} f(t,i,a,\mu_t)\mu_t^i + \frac{\partial V}{\partial t}(t,\mu_t) + \nabla_m V(t,\mu_t)\cdot \frac{d}{dt}\mu_t(t,a)\right)\\
&= \frac{\partial V}{\partial t}(t,\mu_t) + \inf_{a\in[0,M]^d}\left(\sum_{i\in[\![d]\!]}\mu_t^i f(t,i,a,\mu_t) + \sum_{i,j\in[\![d]\!]}\mu_t^i Q_{ij}(t,a^i,\mu_t)\partial_{m_j-m_i}V(t,\mu_t)\right),
\end{aligned}$$

where in the last step, we utilize the dynamics of the system from above and recall that on the interior of the simplex, only derivatives in the directions $(e_j - e_i)_{i,j\in[\![d]\!]}$ are considered in [2]. This last step is justified in [2, Section 2], where the simplex is represented as a $(d-1)$-dimensional submanifold of $\mathbb{R}^d$ via the obvious local chart. Rearranging and notationally replacing μ_t with m, we obtain the HJB equation

$$-\frac{\partial V}{\partial t}(t,m) + \sup_{a\in[0,M]^d}\left(-\sum_{i\in[\![d]\!]} m_i f(t,i,a,m) - \sum_{i,j\in[\![d]\!]} m_i Q_{ij}(t,a^i,m)\partial_{m_j-m_i}V(t,m)\right) = 0.$$

Finally, using more notation from [2], we write $D^i_j V(t,m) = \partial_{m_j-m_i}V(t,m)$ so that the we can write the second term above in terms of the relevant pre-Hamiltonian:

$$\begin{aligned}
\sum_{i\in[\![d]\!]} m_i\left(-\langle Q_{i,\bullet}(t,a,m), D^iV(t,m)\rangle - f(t,i,a,m)\right) = &-\sum_{i\in[\![d]\!]} m_i f(t,i,a,m)\\
&-\sum_{i,j\in[\![d]\!]} m_i Q_{ij}(t,a^i,m)\partial_{m_j-m_i}V(t,m).
\end{aligned}$$

In turn, we arrive at the final HJB equation for the MFCP, given by

$$-\frac{\partial V}{\partial t}(t,m) + \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^iV(t,m)) = 0,$$

from the definition of the Hamiltonian associated with the above pre-Hamiltonian. As before, the associated boundary condition comes directly from the terminal cost of the problem, and is given by

$$V(T,m) = \sum_{i\in[\![d]\!]} m_i g^i(m).$$

This concludes the derivation of the HJB equation for the MFCP. We obtain uniqueness and the regularity of V from Proposition 2.2.1. $\square$

Chapter 3

Convergence of the DGM

In this section, we present our modified DGM algorithm to solve Equation (1.1) and the corresponding convergence proof. We begin by presenting our version of the DGM algorithm in Section 3.1. Next, in Section 3.2, we describe the relevant results from universal approximation theory in addition to proving that the DGM can approximate the solution to Equation (1.1) arbitrarily well in Theorem 3.2.4. In Section 3.3, we present our main convergence results in Theorem 3.3.2. Finally, in Section 3.4, we describe the difficulties in establishing convergence for the algorithm with L^2-loss and analyze in more detail why the proof technique in [1] is invalid for Equation (1.1). We employ the following outline to obtain our existence and convergence results:

(1) By an appropriate version of the universal approximation theorem, we can arbitrarily approximate $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$ with neural network approximators; see Proposition 3.2.1. Specifically, there exists a sequence of neural network approximators $\{\varphi(t,m;\theta^n)\}_{n\in\mathbb{N}}$ such that the DGM loss applied to the sequence converges to zero as $n \to \infty$; see Theorem 3.2.4.

(2) If the DGM loss goes to zero along a sequence of neural network approximators $\{\varphi(t,m;\theta^n)\}_{n\in\mathbb{N}}$, then $\varphi(\cdot,\cdot;\theta^n) \to V(\cdot,\cdot)$ uniformly on $[0,T] \times S_d$; see Theorem 3.3.2.

Throughout this section, we utilize the following operator, defined for $\phi \in \mathcal{C}^{1,1}([0,T]\times S_d)$:

$$\mathcal{L}[\phi](t,m) := -\partial_t \phi(t,m) + \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^i\phi(t,m)). \tag{3.1}$$

Note that the first line in Equation (1.1) can be written as $\mathcal{L}[V](t,m) = 0$.

3.1 DGM Algorithm

In this section, we present a modification of the DGM algorithm, first proposed by [1], in the context of the HJB equation for the MFCP. The DGM aims to efficiently approximate a solution to the above equation using a deep learning-based approach. Specifically, the method learns model parameters $\theta \in \mathbb{R}^P$, where P depends on the dimension d of the simplex S_d, the number of layers in the neural network in use, and the number of nodes in each layer of the neural network. The DGM learns the model parameters θ by minimizing an objective functional, referred to as the *DGM loss*, and given by

$$L(\theta) := \max_{(t,m)\in[0,T]\times S_d} |\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t,m)| + \max_{m\in S_d} |\varphi(T,m;\theta) - \sum_{i\in[\![d]\!]} m_i g^i(m)|. \tag{3.2}$$

Occasionally, we refer to this loss as the L^∞-loss. The maxima over $[0,T]\times S_d$ and S_d are approximated by sampling, as demonstrated in the following algorithm. As with the original DGM algorithm with L^2-loss [1], we utilize stochastic gradient descent (SGD) to find the parameter $\theta \in \mathbb{R}^P$ that minimizes the above loss. Note that in the following algorithm, the architecture of the neural network is fixed, and only the parameter θ is updated by SGD. As a proxy for the true loss functional, given a set of K samples $(t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\dots,K} := \{(t^{(j)}, m^{(j)}, p^{(j)}) : j = 1,\dots,K\} \subset [0,T]\times S_d \times S_d$, we define

$$\begin{aligned} G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\dots,K}, \theta) := &\max_{j=1,\dots,K} |\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)})| \\ &+ \max_{j=1,\dots,K} \left| \varphi(T, p^{(j)};\theta) - \sum_{i\in[\![d]\!]} p_i^{(j)} g^i(p^{(j)}) \right|. \end{aligned} \tag{3.3}$$

In practice, the performance of the following algorithm may vary depending on the sample size K at each step; see the discussion below. Additionally, the learning rate schedule $\alpha^{(n)}$ may determine the convergence rate of the algorithm as before. In most cases, using an optimizer such as AdaGrad or Adam will help speed up convergence; see [37] for an overview of best practices when it comes to efficiently implementing algorithms such as the DGM. Finally, instead of using a tolerance $\delta \in (0,1)$ to determine the convergence of the algorithm, one may instead specify a fixed number of SGD iterations to carry out.

Algorithm 1 Uniform DGM

```
Initialize parameters $\theta^{(0)} \in \mathbb{R}^P$
Initialize tolerance $\delta \in (0,1)$
$n \leftarrow 0$
Uniformly sample $(t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K} \in [0,T] \times S_d \times S_d$
**while** $G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta^{(n)}) \geq \delta$ **do**
    Sample $(t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K} \in [0,T] \times S_d$
    $\theta^{(n+1)} \leftarrow \theta^{(n)} - \alpha^{(n)} \nabla_\theta G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta^{(n)})$
    $n \leftarrow n+1$
**end while**
```

Although the DGM algorithm with L^2-loss as defined in [1], seems to work quite well in practice (as we demonstrate later in Section 4), the structure of the PDE in Equation (1.1) prohibits us from using the same argument as [1] to prove convergence of the DGM algorithm with L^2-loss. Instead, by slightly changing the loss function to the L^∞-loss, given in (3.2), which we use to train the neural network approximation in the DGM algorithm, we can prove the convergence of our modified DGM algorithm to the unique value function of the MFCP. By utilizing the above loss functional that approximates the uniform norm of the PDE and terminal condition rather than the squared error of the PDE and the terminal condition, we can leverage the theory of viscosity solutions for first-order HJB equations from [33] in our convergence proof.

In practice, one may compute the L^∞-loss in Algorithm 1 using a smooth maximum that approximates a maximum, but remains differentiable. For instance, a widely-used smooth maximum in the practice of machine learning is *log–sum–exp*, given by

$$f(x_1, \ldots, x_n) := \log \left(\sum_{i=1}^{n} \exp(x_i) \right)$$

Because

$$\exp\left(\max_{i=1,\ldots,n} x_i \right) \leq \sum_{i=1}^{n} \exp(x_i) \leq n \exp(\max_{i=1,\ldots,n} x_i),$$

we obtain the obvious bound

$$\max\{x_1, \ldots, x_n\} \leq \log \left(\sum_{i=1}^{n} \exp(x_i) \right) \leq \max\{x_1, \ldots, x_n\} + \log n,$$

with a strict inequality on the right-hand side unless $x_1 = \ldots = x_n$. For previous work involving stochastic gradient descent applied to a maximized loss function and insights into its robustness, refer to [38, 39]. Empirically, we find that JAX's built-in subgradient calculus is sufficient to implement Algorithm 1 without utilizing a smooth maximum [40].

As noted above, we approximate that maxima over $[0, T] \times S_d$ and S_d respectively by uniformly sampling K points in each region. This incurs some error between the approximate quantities in Equation (3.3) and their counterparts in Equation (3.2). We quantify the error asymptotically in the number of samples K via the following proposition, based on the analogous result for numerically solving the master equation in [14, Lemma 1].

Proposition 3.1.1. *Assume that for any $\theta \in \mathbb{R}^P$, the corresponding neural network approximator $\varphi(\cdot, \cdot; \theta)$ is Lipschitz continuous with Lipschitz derivatives, and all Lipschitz constants are uniformly bounded by some $C > 0$. Then,*

$$\mathbb{E}\left|L(\theta) - G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\dots,K}, \theta)\right| \leq \tilde{C}_d K^{-1/(d-1)},$$

where $L(\theta)$ is the true DGM loss from Equation (3.2), $G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\dots,K}, \theta)$ is the sampled DGM loss for K uniform samples on $[0, T] \times S_d \times S_d$ from Equation (3.3), and $\tilde{C}_d > 0$ is a constant that depends on the class of neural networks used in Algorithm 1 and the dimension d.

Proof. In the following proof, let

$$(t^\star, m^\star) \in \operatorname*{argmax}_{(t,m) \in [0,T] \times S_d} |\mathcal{L}[\varphi(\cdot, \cdot; \theta)](t, m)|$$

and

$$m^\star \in \operatorname*{argmax}_{m \in S_d} \left|\varphi(T, m; \theta) - \sum_{i \in [\![d]\!]} m_i g^i(m)\right|.$$

Note that because $\varphi(\cdot, \cdot; \theta) \in \mathcal{C}^\infty([0, T] \times S_d)$, both of the above maximizers are obtained on the compact set $[0, T] \times S_d$. Similarly, we denote the empirical maximizers by

$$j_1 \in \operatorname*{argmax}_{j=1,\dots,K} |\mathcal{L}[\varphi(\cdot, \cdot; \theta)](t^{(j)}, m^{(j)})|$$

and

$$j_2 \in \underset{j=1,\ldots,K}{\operatorname{argmax}} \left| \varphi(T, p^{(j)}; \theta) - \sum_{i \in [\![d]\!]} p_i^{(j)} g^i(p^{(j)}) \right| .$$

Using the above notation, we can write

$$L(\theta) = |\mathcal{L}[\varphi(\cdot, \cdot; \theta)](t^\star, m^\star)| + \left| \varphi(T, p^\star; \theta) - \sum_{i \in [\![d]\!]} p_{2,i} g^i(p^\star) \right|$$

and

$$G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta) = |\mathcal{L}[\varphi(\cdot, \cdot; \theta)](t^{(j_1)}, m^{(j_1)})| + \left| \varphi(T, p^{(j_2)}; \theta) - \sum_{i \in [\![d]\!]} p_i^{(j_2)} g^i(p^{(j_2)}) \right| .$$

Now, define

$$j_{\min,1} := \underset{j=1,\ldots,K}{\operatorname{argmin}} \left\{ |t^{(j)} - t^\star| \right\}, \tag{3.4}$$

$$j_{\min,2} := \underset{j=1,\ldots,K}{\operatorname{argmin}} \left\{ \|m^{(j)} - m^\star\|_2 \right\}, \tag{3.5}$$

$$j_{\min,3} := \underset{j=1,\ldots,K}{\operatorname{argmin}} \left\{ \|p^{(j)} - p^\star\|_2 \right\}, \tag{3.6}$$

allowing us to describe the sampled points the sampled point closest to the true maximizer $(t^\star, m^\star, p^\star)$. Now, we evidently have that

$$0 \leq L(\theta) - G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta) \leq L(\theta) - G(t^{(j_{\min})}, m^{(j_{\min})}, p^{(j_{\min})}, \theta).$$

Now, we leverage the uniform Lipschitz continuity of $\varphi(\cdot, \cdot; \theta)$, as well as Assumption 2.1, to obtain a Lipschitz bound of the form

$$\begin{aligned} &|L(\theta) - G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta)| \\ &\quad \leq |L(\theta) - G(t^{(j_{\min,1})}, m^{(j_{\min,2})}, p^{(j_{\min,3})}, \theta)| \\ &\quad \leq C(|t^{(j_{\min,1})} - t^\star| + \|m^{(j_{\min,2})} - m^\star\|_2 + \|p^{(j_{\min,3})} - p^\star\|_2), \end{aligned} \tag{3.7}$$

where C depends on the Lipschitz constants of the Hamiltonian H (as discussed in Lemma 3.2.3 below, Assumption (A) ensures that H is Lipschitz), the terminal condition G, and the uniform Lipschitz bound on neural network approximators (and their derivatives) introduced in the assumptions of the proposition. Indeed, we observe that by the reverse triangle inequality,

$$\begin{aligned}
&|\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^\star, m^\star)| - |\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j_{\min,1})}, m^{(j_{\min,2})})| \\
&\quad \leq |\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^\star, m^\star) - \mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j_{\min,1})}, m^{(j_{\min,2})})| \\
&\quad \leq C(|t^\star - t^{(j_{\min,1})}| + \|m^\star - m^{(j_{\min,2})}\|_2).
\end{aligned}$$

In the last line, we use the fact that

$$\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t,m) = -\partial_t \varphi(t,m;\theta) + \sum_{i\in[\![d]\!]} m_i H^i(t, m, D^i\varphi(t,m;\theta))$$

is Lipschitz continuous with Lipschitz constant C (taking $C > 0$ larger if necessary), as it is simply the sum and composition of Lipschitz functions under our assumptions. Repeating the exact same reasoning on the terminal cost term, which is of the form

$$p \mapsto \left| \varphi(T,p;\theta) - \sum_{i\in[\![d]\!]} p_i g^i(p) \right|$$

and is therefore also Lipschitz continuous with constant $C > 0$, we obtain Equation (3.7).

Consequently, we can bound the expected value of each of the three terms in Equation (3.7), with the expectation taken with respect to a uniformly-random sample of K points from $[0,T] \times S_d \times S_d$. For the first term, observe that by the definition of $j_{\min}$ in Equation (3.4) above, we have that

$$\mathbb{E}|t^{(j_{\min,1})} - t^\star| = \mathbb{E}\left[\min_{j=1,\dots,K} |t^{(j)} - t^\star|\right] \leq \mathbb{E}\left[\min_{j=1,\dots,K} t^{(j)}\right]. \tag{3.8}$$

The inequality in the preceding display follows because $t^{(j)} \in [0,T]$ are sampled uniformly, so for any $j \in [\![K]\!]$ and $\delta \in (0,1)$, we see that

$$\mathbb{P}(|t^{(j)} - t^\star| > \delta T) \leq 1 - 2\delta \leq 1 - \delta = \mathbb{P}(t^{(j)} > \delta T).$$

In turn,

$$\mathbb{P}(|t^{(j_{\min,1})} - t^\star| \leq \delta T) = 1 - \prod_{j=1}^{K} \mathbb{P}(|t^{(j)} - t^\star| > \delta T) \geq 1 - \prod_{j=1}^{K} \mathbb{P}(t^{(j)} > \delta T)$$
$$= \mathbb{P}\left(\min_{j=1,\dots,K} t^{(j)} \leq \delta T\right)$$

for all $\delta \in (0,1)$. It then follows that the non-negative random variable $\min_{j=1,\dots,K} t^{(j)}$ stochastically dominates $|t^{(j_{\min,1})} - t^\star|$, in which case an argument from elementary probability theory implies the inequality in Equation (3.8) above. Now, because

$$\mathbb{P}\left(\min_{j=1,\dots,K} t^{(j)} \leq t\right) = 1 - \left(1 - \frac{T-t}{T}\right)^K$$

for $t \in [0,T]$, we observe that $\min_{j=1,\dots,K} t^{(j)}$ in fact has a density given by

$$f(t) = \frac{K}{T}\left(\frac{T-t}{T}\right)^{K-1}.$$

This allows us to directly compute

$$\mathbb{E}\left[\min_{j=1,\dots,K} t^{(j)}\right] = \int_0^T \frac{Kt}{T}\left(\frac{T-t}{T}\right)^{K-1} dt = TK\int_0^1 (1-s)s^{K-1} ds$$
$$= \frac{T}{1+K}.$$

With this bound in mind, we turn towards bounding the last two terms in Equation (3.7) above. Because both $m^{(j)}$ and $p^{(j)}$ are sampled uniformly from the simplex S_d, it suffices to bound the second term, as we can then bound the last term analogously. The following argument is identical to that of [14, Lemma 1], but we include it here for the sake of completeness. First, observe that for any $m \in S_d$, we can write

$$m_d = 1 - \sum_{i \in [\![d-1]\!]} m_i.$$

As a result, it follows that

$$\begin{aligned}
\|m^{(j_{\min,2})} - m^\star\|_2^2 &= \sum_{i \in \llbracket d \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)^2 \\
&= \sum_{i \in \llbracket d-1 \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)^2 + \left(1 - \sum_{i \in \llbracket d-1 \rrbracket} m_i^{(j_{\min,2})} - 1 + \sum_{i \in \llbracket d-1 \rrbracket} m_i^\star\right)^2 \\
&= \sum_{i \in \llbracket d-1 \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)^2 + \left(\sum_{i \in \llbracket d-1 \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)\right)^2 .
\end{aligned}$$

In turn, by the Cauchy–Schwarz, inequality, we can bound the preceding quantity above by

$$\sum_{i \in \llbracket d-1 \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)^2 + (d-1) \sum_{i \in \llbracket d-1 \rrbracket} (m_i^{(j_{\min,2})} - m_i^\star)^2 = d\|\hat{\pi}(m^{(j_{\min,2})} - m_i^\star)\|_2^2, \tag{3.9}$$

where $\hat{\pi} : \mathbb{R}^d \to \mathbb{R}^{d-1}$ is the projection of S_d onto $\mathbb{R}^{d-1}$; see Section 3.3 below. Now, observe that

$$\begin{aligned}
\mathbb{E}[\|\hat{\pi}(m^{(j_{\min,2})} - m_i^\star)\|_2] &= \mathbb{E}\left[\min_{j=1,\dots,K} \|\hat{\pi}(m^{(j)} - m^\star)\|_2\right] \\
&\le \mathbb{E}\left[\min_{j=1,\dots,K} \|\hat{\pi}(m^{(j)})\|_2\right] \\
&\le \mathbb{E}\left[\min_{j=1,\dots,K} \|\hat{\pi}(m^{(j)})\|_1\right] .
\end{aligned}$$

To obtain the above bound, note that the case in which $\hat{\pi}(m^\star) = 0$ provides an upper bound, by analogous reasoning as in Equation (3.8). Additionally, we apply the fact that for any $v \in \mathbb{R}^{d-1}$, $\|v\|_2 \le \|v\|_1$. Now, because $m^{(j)}$ is sampled uniformly from S_d, [41, Remark 1.3] implies that if $X_1, \dots, X_d \sim \text{Exp}(1)$, then

$$m_i^{(j)} = \frac{X_i}{\sum_{\ell \in \llbracket d \rrbracket} X_\ell}.$$

In other words, to sample a point uniformly on the simplex S_d, one can instead sample d i.i.d. exponential random variables and normalize by their sum to obtain each coordinate of a uniformly sampled point on S_d. In turn, we see that

$$\|\hat{\pi}(m^{(j)}\|_1 = \sum_{i\in[\![d-1]\!]} m_i^{(j)} = \frac{\sum_{i\in[\![d-1]\!]} X_i}{\sum_{i\in[\![d]\!]} X_i} \sim \beta(d-1,1),$$

as the sum of exponential random variables satisfies

$$\sum_{i\in[\![d-1]\!]} X_i \sim \Gamma(d-1,1), \quad \sum_{i\in[\![d]\!]} X_i \sim \Gamma(d,1),$$

respectively, and the ratio of two gamma distributions, distributed as $\Gamma(\alpha_1,1)$ and $\Gamma(\alpha_2,1)$ respectively, follows a beta distribution, distributed as $\beta(\alpha_1,\alpha_2)$. From this characterization, we can directly bound the expectation of $\min_{j=1,\ldots,K}\|\pi(m^{(j)}\|_1$ by bounding the expectation of the minimum of K i.i.d. random variables, each distributed as $\beta(d-1,1)$. Fortunately, the cumulative distribution function of such a random variable has a relatively simple description:

$$\begin{aligned}\mathbb{P}\left(\min_{j=1,\ldots,K}\|\pi(m^{(j)}\|_1 \le x\right) &= 1-\prod_{j=1}^{K}(1-\mathbb{P}(\|\pi(m^{(j)}\|_1 \le x))\\ &= 1-(1-x^{d-1})^K,\end{aligned}$$

using the well-known cumulative distribution of a random variable distributed as $\beta(d-1,1)$. From this, and the fact that $\|\pi(m^{(j)}\|_1 \in [0,1]$, we obtain

$$\begin{aligned}\mathbb{E}\left[\min_{j=1,\ldots,K}\|\pi(m^{(j)}\|_1\right] &= \int_0^1 \mathbb{P}\left(\min_{j=1,\ldots,K}\|\pi(m^{(j)}\|_1 \ge x\right)dx\\ &= \int_0^1 (1-x^{d-1})^K dx\\ &= \frac{1}{d-1}\int_0^1 s^{1/(d-1)-1}(1-s)^K ds,\end{aligned}$$

taking $s = x^{d-1}$ in the last equality. The integrand in the last line above, however, is the (unnormalized) probability density function of a random variable distributed as $\beta(1/(d-1),K+1)$. The corresponding normalization is well-known, and it is given by

$$\int_0^1 s^{1/(d-1)-1}(1-s)^K ds = \frac{\Gamma(1/(d-1))\Gamma(K+1)}{\Gamma(K+1+1/(d-1))},$$

where Γ is the standard Gamma function. Taking

$$C_d := \frac{\Gamma(1/(d-1))}{d-1}, \tag{3.10}$$

we utilize the bound on the ratio of Gamma functions, first presented in [42] and based on the standard Stirling approximation for the Gamma function, to see that

$$\int_0^1 s^{1/(d-1)-1}(1-s)^K ds = C_d K^{-1/(d-1)} \left[1 - \frac{2(d-1)-1}{2K(d-1)} + O(K^{-2})\right].$$

Returning to Equation (3.9), we see that

$$\begin{aligned}\mathbb{E}[\|\|\|m^{(j_{\min,2})} - m^\star\|_2] &\leq \sqrt{d}\mathbb{E}\left[\min_{j=1,\ldots,K} \|\hat{\pi}(m^{(j)})\|_1\right] \\ &\leq \sqrt{d}C_d K^{-1/(d-1)} \left[1 - \frac{2(d-1)-1}{2K(d-1)} + O(K^{-2})\right].\end{aligned}$$

Repeating the exact same reasoning on the third term in Equation (3.7), we similarly obtain

$$\mathbb{E}[\|\|\|p^{(j_{\min,2})} - p^\star\|_2] \leq \sqrt{d}C_d K^{-1/(d-1)} \left[1 - \frac{2(d-1)-1}{2K(d-1)} + O(K^{-2})\right].$$

Finally, we have that

$$\mathbb{E}|t^{(j_{\min,1})} - t^\star| = \frac{T}{1+K}$$

from Equation (3.8) above. Substituting all three quantities into Equation (3.7) yields an estimate of the form

$$|L(\theta) - G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta)| \leq \sqrt{d}CC_d\mathcal{O}(K^{-1/(d-1)}), \tag{3.11}$$

where C depends on the Lipschitz bound on the class of neural networks used in Algorithm 1 and C_d is given in Equation (3.10). Taking $\tilde{C}_d := \sqrt{d}CC_d$ completes the proof. □

As a result of the above error bound, by taking the sample size K sufficiently large, we can approximate the true loss in Equation (3.2) via the empirical, sampled loss in

Equation (3.3). Consequently, if the sampled loss converges to zero, then so must the true loss, with the gap between the two described by the error term in Proposition 3.1.1. We stress that the above bound is asymptotic. Rearranging the bound reveals that if we wish to approximate the true DGM loss by the sampled loss, up to an error of $\varepsilon > 0$, then we require $K = \mathcal{O}\left(\varepsilon^{1-d}\right)$ samples. Disappointingly, this is still exponential in the dimension d of the HJB equation in question. However, the above asymptotic bound does not necessarily mean that an exponentially large number of samples will be necessary to accurately approximate the DGM loss. In practice, the experiments in Section 4 indicate that the upper bound above is likely pessimistic, as taking $K \approx 10000$ is typically sufficient to obtain arbitrarily small loss after running SGD, measured by the average sampled loss over *many* resamples of K points on $[0, T] \times S_d \times S_d$. Empirically, this phenomenon appears to hold up to at least $d = 200$. Furthermore, we explore applications of adversarial training, a technique that is specifically tailored towards accurately approximating L^∞-loss by biasing the algorithm towards samples that result in large L^∞-loss, in Section 4.2.

3.2 Universal Approximation Via Neural Networks

In this section, we provide a brief, non-comprehensive review of universal approximation theorems, with the goal of summarizing results relevant to the DGM. By a "two-layer" feedforward neural network, we refer to a neural network with one input layer and *one* hidden layer as defined by (3.12), taken with $L = 1$. Although such networks are relatively simple compared modern deep neural networks, their universal approximation power is sufficient for the purposes of the convergence result in this section. Theorem 3.2.1 is just one of many results that describe the approximation power of this class of neural networks; see [43] for a similar survey, albeit with a focus on implementation.

Universal approximation results relevant to this paper fall broadly into two categories: approximations in L^p spaces for $p < \infty$ (or in $W^{k,p}$ spaces if the approximation of derivatives is also desired) and approximations in $\mathcal{C}^k$ for $k \geq 0$. Modern universal approximation results largely stem from those established by Hornik, Stinchcombe, White, and Cybenko in the early 1990s. In particular, the former three authors first showed that neural networks with (possibly discontinuous) "squashing function" activations are uniformly dense on compact sets in $\mathcal{C}(\mathbb{R})$, with respect to the uniform norm [44]. Cybenko then showed that neural networks with sigmoidal activations share the same approximation power [45]. Hornik, Stinchombe, and White then extended the result of Cybenko to neural networks with non-sigmoidal (indeed, bounded and nonconstant) activations, also providing estimates of the derivatives of an unknown function $f \in W^{k,p}(\Omega)$ for $k \geq 1$ and $p \geq 1$, with $\Omega \subset \mathbb{R}^n$ compact [46]. Theorem 3.2.1 is due [47], who subsequently showed that functions with k continuous derivatives can be uniformly approximated on compact sets in $\mathcal{C}^k$-norm in addition to the standard norm on $W^{k,p}$ [47]. Out of the universal approximation theorems shown in the early 1990s, Hornik's result is the most general. However, his proof, relying on the Hahn–Banach theorem, is not constructive and does not extend to neural networks with bounded weights.

Conversely, Stinchcombe and White showed that continuous functions can be approximated, in uniform norm, by neural networks with bounded weights using an argument based on the Stone–Weierstrass theorem [48]. In Section 3.4, we discuss networks with bounded weights in more detail, as well as the impact that bounding weights may have on approximation power. Indeed, in order to establish equicontinuity as in Theorem 3.4.3, we require approximation by a sequence of neural networks with bounded, summable weights. In the case of bounded neural networks, however, approximation of a function *and* its derivatives does not seem to be covered by the current literature.

Although the vast majority of results concerning universal approximation by feedforward neural networks were stated and proved in the early 1990s, several authors have made recent attempts to extend classical results to more complicated architectures, more general convergence guarantees, and provide explicit constructions of universal approximators. For instance, Mhaskar and Micchelli show in [49] that functions and their derivatives can be uniformly approximated in $L^p([-1,1]^d)$ for $p \geq 1$ using a constructive, Fourier-analytic approach via approximation by periodic functions.

Additionally, there has been recent interest in exploring the approximation power of neural networks with specific activation functions such as hyperbolic tangent neural networks. For example, [50] considers the approximation power of tanh neural networks in the standard norm on $W^{k,p}(\Omega)$ for a compact domain $\Omega \subset \mathbb{R}^d$, deducing asymptotic bounds on the weights of the neural network and much more explicit bounds on the approximation power of a network with a fixed number of hidden units. In a similar vein,

[51] derive promising results for networks with piecewise quadratic activations (specifically ReQU neural networks), demonstrating that ReQU networks with bounded weights *can* approximate functions and their derivatives in Hölder norms. In particular, [51] shows that if $f \in \mathcal{C}^{2,\alpha}([0,1]^d)$ for $\alpha \in (0,1]$, then for any $\varepsilon > 0$, there exists a *deep* neural network φ_f with weights in $[-1,1]$ such that $\|f - \varphi_f\|_{\mathcal{C}^{2,\alpha}([0,1]^d)} < \varepsilon$. This construction, however, is not currently considered in the proof of Theorem 3.2.4, as we limit ourselves to smooth activation functions here.

More concretely, we utilize the classical universal approximation result of Hornik [47] to establish that the uniform DGM loss can be made arbitrarily small by neural network approximators in Theorem 3.2.4. Then, we lay the foundations for our main convergence result via Corollary 3.2.5, which provides a reformulation of the result in Theorem 3.2.4 that is better suited for the language of viscosity solutions.

Although all universal approximation results in [47] hold for two-layer neural networks (i.e., a single hidden layer), we allow neural networks with multiple layers, including deep neural networks with modern architectures. In general, a network with L layers, maximum width n, and a common activation function σ takes the form

$$\varphi(t,m;\theta) := \sigma(W_L \dots \sigma(W_1 m + \alpha t + c_1) \dots + c_L), \tag{3.12}$$

where the activation function σ is applied elementwise. Above, W_i are weight matrices, c_i are bias vectors, and α is a scalar weight. In turn, the parameters of each neural network are of the form $\theta = (W_1, \dots, W_L, c_1, \dots, c_L, \alpha) \in \mathbb{R}^P$ (upon flattening all weight matrices into vectors), where P depends on the maximum width of the network, the depth L of the network, and the dimension d of the simplex S_d. In turn, we take $\mathfrak{C}^{(P)}_{d+1}(\sigma)$ to be the class of neural networks with parameters θ of dimension at most P (but any number of layers L), from which we define

$$\mathfrak{C}_{d+1}(\sigma) := \bigcup_{P=1}^{\infty} \mathfrak{C}^{(P)}_{d+1}(\sigma).$$

This is a slight departure from the notation of [47], but all the relevant universal approximation results therein still hold in this more general context. With the above notation in mind, we can apply the universal approximation theorem [47, Theorem 3], stated as follows:

Proposition 3.2.1. *If $\sigma \in \mathcal{C}^m(\mathbb{R})$ is nonconstant and bounded, then $\mathfrak{C}_{d+1}$ is uniformly m-dense on compact sets in $\mathcal{C}^m(\mathbb{R}^{d+1})$. In particular, for all $h \in \mathcal{C}^m(\mathbb{R}^{d+1})$, all compact subsets $K \subset \mathbb{R}^{d+1}$, and any $\varepsilon > 0$, there exists $\psi = \psi(h, K, \varepsilon) \in \mathfrak{C}_{d+1}$ such that $\|h - \psi\|_{\mathcal{C}^m(K)} < \varepsilon$.*

In the implementation in Section 4, we take $\sigma(y) = \tanh(y)$, a typical choice of activation function that is smooth, nonconstant, and bounded, and therefore satisfies all the criteria of Proposition 3.2.1. Note also that with this choice of σ, any element of $\mathfrak{C}_{d+1}(\sigma)$ is smooth (as a linear combination of smooth functions), ensuring that any $\varphi \in \mathfrak{C}_{d+1}(\sigma)$ has Lipschitz-continuous first derivative. See [50] for further justification of this choice of activation function in terms of the approximation guarantees that it brings.

Remark 3.2.2. In Proposition 3.1.1, we required that the neural networks used to approximate the unique classical solution $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$ to Equation (1.1) were *uniformly* Lipschitz continuous with uniformly Lipschitz derivatives. Because the value function $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$ is itself Lipschitz continuous with Lipschitz derivative, this is reasonable assumption to impose, and we do not lose any of the approximation power provided by Proposition 3.2.1 above. Indeed, recent work shows that neural networks often possess the Lipschitz continuity properties assumed in Proposition 3.1.1 after training, and imposing such Lipschitz constraints on neural networks improves generalization performance; see [52, 53, 54].

With the above background in mind, we move towards approximating solutions to the HJB equation for the MFCP (1.1) using the DGM. Through Assumptions (A) – (C), we clarify several useful properties for proving convergence of the DGM to the solution of the HJB equation.

First, recall that by Proposition 2.2.1, Equation (1.1) admits a unique classical solution $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$. Second, under Assumption (A), the PDE-Hamiltonian in Equation (1.1) is Lipschitz continuous. In particular, $H(t,m,p) = \sum_{i \in [\![d]\!]} m_i H^i(t,m,p)$, is Lipschitz continuous so that for any $(t_1, m^{(1)}, p_1), (t_2, m^{(2)}, p_2) \in [0,T] \times S_d \times S_d$, we have that

$$\begin{aligned}
&\left|\sum_{i\in[\![d]\!]}(m_i^{(1)}-m_i^{(2)})(H^i(t_1,m^{(1)},p_1)-H^i(t_2,m^{(2)},p_2))\right| \\
&\quad\le \kappa\|(t_1-t_2,m^{(1)}-m^{(2)},p_1-p_2)\|_2 \\
&\quad\le \kappa(\|t_1-t_2\|_2+\|m^{(1)}-m^{(2)}\|_2+\|p_1-p_2\|_2)
\end{aligned}$$

for some constant $\kappa > 0$, where $\|\cdot\|_2$ denotes the Euclidean norm. In fact, we only need the following lemma, which also follows from Assumption (A).

Lemma 3.2.3. *For each $i \in [\![d]\!]$ and fixed $(t,m) \in [0,T]\times S_d$, the map $p \mapsto H^i(t,m,p)$ is Lipschitz continuous with a common Lipschitz constant $C > 0$ that does not depend on i.*

Proof. By the definition of the Hamiltonian provided in (1.18), we have that

$$\begin{aligned}
H^i(t,m,p)-H^i(t,m,p') &\le \sup_{a_k\in[0,M],\,k\in[\![d]\!]\setminus\{i\}}\left(-\sum_{k\in[\![d]\!]\setminus\{i\}}a_kp_k-f(t,i,a,m)\right) \\
&\quad-\sup_{a_k\in[0,M],\,k\in[\![d]\!]\setminus\{i\}}\left(-\sum_{k\in[\![d]\!]\setminus\{i\}}a_kp'_k-f(t,i,a,m)\right) \\
&=\sup_{a_k\in[0,M],\,k\in[\![d]\!]}\left|\sum_{k\in[\![d]\!]\setminus\{i\}}a_k(p_k-p'_k)\right| \\
&\le M\left|\sum_{k\in[\![d]\!]}(p_k-p'_k)\right| \\
&\le \sqrt{d}M\|p-p'\|_2,
\end{aligned}$$

where M is a bound on transition rates introduced in Section 2. In the last line above, we use the fact that for any $v\in\mathbb{R}^d$, $\|v\|_1 \le \sqrt{d}\|v\|_2$. Switching the roles of p and p' above and taking $C=\sqrt{d}M$ completes the proof. □

We show below that by utilizing the uniform error, given in Equation (3.2) we can obtain the desired convergence result. With the ultimate goal of showing that a neural network can approximate the value function $V(t,m)$ on $[0,T]\times S_d$ arbitrarily well in the uniform norm by taking the number of neurons in the network sufficiently large, we first show the following existence result. This result in fact holds for both the our DGM loss functional and the DGM with L^2-loss, as shown in Section 3.4. In particular, the following theorem establishes the existence of a sequence of neural networks that makes the DGM loss arbitrarily small.

Theorem 3.2.4. *Let $\sigma \in \mathcal{C}^1(\mathbb{R})$ be bounded and nonconstant. For every $\varepsilon > 0$, there exists a constant $\tilde{K}(d, T, C) > 0$, where d is the dimension of the simplex S_d, T is the finite time horizon of the MFCP, and C is the Lipschitz constant of the PDE-Hamiltonian in Equation* (1.1)*, such that for some $\varphi = \varphi(\cdot, \cdot; \theta) \in \mathfrak{C}_{d+1}(\sigma)$, the DGM loss functional in Equation* (3.2) *satisfies $L(\theta) \leq \tilde{K}\varepsilon$.*

Proof. Note that $\Omega_T := [0, T] \times S_d$ is a compact set in $\mathbb{R}^{d+1}$. Thus, by Proposition 3.2.1 above, we know that for the unique solution $V \in \mathcal{C}^{1,1}(\Omega_T)$ to Equation (1.1) and any $\varepsilon > 0$, there exists $\varphi \in \mathfrak{C}_{d+1}(\sigma)$ such that

$$\begin{aligned}\varepsilon > \sup_{(t,m)\in\Omega_T} |V(t,m) - \varphi(t,m;\theta)| + \sup_{(t,m)\in\Omega_T} |\partial_t V(t,m) - \partial_t \varphi(t,m;\theta)| \\ + \sup_{(t,m)\in\Omega_T} |\nabla_m V(t,m) - \nabla_m \varphi(t,m;\theta)|\end{aligned} \tag{3.13}$$

For such $\varphi \in \mathfrak{C}_{d+1}(\sigma)$, the Lipschitz continuity of the PDE-Hamiltonian in Equation (1.1), due to Lemma 3.2.3, yields

$$\begin{aligned}&\left| \sum_{i\in[\![d]\!]} m_i H^i(t, m, D^i\varphi(t,m;\theta)) - \sum_{i\in[\![d]\!]} m_i H^i(t, m, D^i V(t,m)) \right| \\ &\quad \leq C \sum_{i\in[\![d]\!]} |D^i\varphi(t,m;\theta) - D^i V(t,m)|,\end{aligned}$$

where C is the Lipschitz constant of the PDE-Hamiltonian. Above, we use the fact that $|m_i| \leq 1$ for any $m \in S_d$. Next, by denoting the standard basis of $\mathbb{R}^d$ by $\{e_i\}_{i\in[\![d]\!]}$, we observe that for each $i \in [\![d]\!]$ and any $(t, m) \in \Omega_T$,

$$\begin{aligned}|D^i\varphi(t,m;\theta) - D^i V(t,m)|^2 &= \frac{1}{2}\sum_{j=1}^{d} |(\nabla_m\varphi(t,m;\theta) - \nabla_m V(t,m)) \cdot (e_j - e_i)|^2 \\ &\leq \frac{1}{2}\sum_{j=1}^{d} |\nabla_m V(t,m) - \nabla_m\varphi(t,m;\theta)|^2 |e_j - e_i|^2 \\ &\leq d|\nabla_m V(t,m) - \nabla_m\varphi(t,m;\theta)|^2,\end{aligned} \tag{3.14}$$

applying the Cauchy–Schwarz inequality in the second line above. Putting both bounds together, we obtain

$$\begin{aligned}
&\left|\sum_{i\in[\![d]\!]} m_i H^i(t,m,D^i\varphi(t,m;\theta)) - \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^iV(t,m))\right| \\
&\quad \leq Cd^{3/2}|\nabla_m V(t,m) - \nabla_m\varphi(t,m;\theta)|.
\end{aligned} \tag{3.15}$$

Now, let $\tilde{K} = Cd^{3/2} > 0$. We conclude by noting that the value function V satisfies $\mathcal{L}[V](t,m) = 0$ for all $(t,m) \in \Omega_T$ (in addition to the terminal condition of Equation (1.1)), which allows us to conclude that for $\varphi = \varphi(\cdot,\cdot;\theta)$,

$$\begin{aligned}
L(\theta) &= \max_{(t,m)\in\Omega_T} |\mathcal{L}[\varphi](t,m)| + \max_{m\in S_d}\left|\varphi(T,m;\theta) - \sum_{i\in[\![d]\!]} m_i g^i(m)\right| \\
&= \max_{(t,m)\in\Omega_T} |\mathcal{L}[\varphi](t,m) \; - \mathcal{L}[V](t,m)| + \max_{m\in S_d}|\varphi(T,m;\theta) - V(T,m)| \\
&\leq \max_{(t,m)\in\Omega_T}\left| m_i H^i(t,m,D^i\varphi(t,m;\theta)) - \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^iV(t,m))\right| \\
&\quad + \max_{(t,m)\in\Omega_T} |\partial_t V(t,m) - \partial_t\varphi(t,m;\theta)| \\
&\quad + \max_{m\in S_d}|\varphi(T,m;\theta) - V(T,m)| \\
&\leq \tilde{K}\varepsilon
\end{aligned}$$

by applying the approximation result from (3.13) and taking $\tilde{K} > 0$ larger if necessary. Note that the constant $\tilde{K}$ now depends on d, C, and T as claimed. □

The following corollary, which also relates to the existence of an approximating sequence of neural networks, utilizes the same universal approximation theorem as Theorem 3.2.4. In particular, we can obtain a sequence of neural networks that satisfies a corresponding sequence of PDEs, with a measurable error term that uniformly converges to zero.

Corollary 3.2.5. *There exists a sequence of parameters $\{\theta^n\}_{n\in\mathbb{N}}$, such that $\varphi^n \to V$ uniformly as $n \to \infty$, where V is the unique classical solution to Equation* (1.1)*, and*

$\varphi^n(t,m) := \varphi(t,m;\theta^n)$. *Given such* φ^n, *we define* $e^n : [0,T] \times S^d \to \mathbb{R}$ *as follows:*

$$\begin{aligned} e^n(t,m) &:= \mathcal{L}[\varphi^n](t,m), \qquad t \in [0,T), \\ e^n(T,m) &:= \varphi^n(T,m) - \sum_{i \in [\![d]\!]} m_i g^i(m). \end{aligned} \tag{3.16}$$

Then, $\|e^n\|_\infty \to 0$ *as* $n \to \infty$*, noting that each* e^n *depends on the neural network parameter* θ^n.

Proof. From Proposition 3.2.1, we know that there exists a sequence of neural networks $\{\varphi^n\}_{n\in\mathbb{N}} \subset \mathfrak{C}_{d+1}(\sigma) \subset \mathcal{C}^{1,1}([0,T] \times S_d)$, parametrized by a set of parameters $\{\theta^n\}_{n\in\mathbb{N}}$, such that (3.13) holds for $\varepsilon = n^{-1}$. Thus, it immediately follows that $\|\varphi^n - V\|_\infty \to 0$ as $n \to \infty$, yielding a sequence φ^n of neural networks parametrized by θ^n that converges uniformly to V, the classical solution to Equation (1.1). By construction, φ^n satisfies the PDE in (3.16). Furthermore, because each $e^n : [0,T] \times S_d \to \mathbb{R}$ is continuous on both $[0,T) \times S_d$ and $\{T\} \times S_d$, it is evidently measurable.

Finally, to see that $\|e^n\|_\infty \to 0$ as $n \to \infty$, we reuse many of the estimates from the proof of Theorem 3.2.4. In particular, we can write

$$\begin{aligned} e^n(t,m) &= -\partial_t \varphi^n(t,m) + \sum_{i \in [\![d]\!]} m_i H^i(t,m,D^i\varphi^n(t,m)) \\ &= \partial_t V(t,m) - \partial_t \varphi^n(t,m) + \sum_{i \in [\![d]\!]} m_i \left[H^i(t,m,D^i\varphi^n(t,m)) - H^i(t,m,D^iV(t,m)) \right], \end{aligned}$$

for $(t,m) \in [0,T) \times S_d$, using the fact that V solves Equation (1.1). By the fact that $m \in S_d$ and the Lipschitz continuity of H^i for each $i \in [\![d]\!]$, we again have that

$$\begin{aligned} \left| \sum_{i \in [\![d]\!]} m_i \left[H^i(t,m,D^i\varphi^n(t,m)) - H^i(t,m,D^iV(t,m)) \right] \right| &\le C \sum_{i \in [\![d]\!]} |D^i\varphi^n(t,m) - D^iV(t,m)| \\ &\le 2dC |\nabla_m V(t,m) - \nabla_m \varphi^n(t,m)|. \end{aligned}$$

Thus, we have that for all $(t,m) \in [0,T) \times S_d$

$$|e^n(t,m)| \le 2dC|\nabla_m V(t,m) - \nabla_m \varphi^n(t,m)| + |\partial_t V(t,m) - \partial_t \varphi^n(t,m)| \le \frac{2dC}{n} + \frac{1}{n},$$

by the construction of φ^n from (3.13). Similarly, at the terminal time $t = T$, observe that

$$|e^n(T,m)| = \left|\varphi^n(T,m) - \sum_{i\in[\![d]\!]} m_i g^i(m)\right| = |\varphi^n(T,m) - V(T,m)| \leq \frac{1}{n}.$$

Thus, it follows that $\|e^n\|_\infty \to 0$ as $n \to \infty$ as claimed. □

Remark 3.2.6. The above sequence of neural networks satisfies several properties based on its construction. Specifically, note that by the converse of the Arzelà–Ascoli theorem, $\{\varphi^n\}_{n\in\mathbb{N}}$ is uniformly bounded and equicontinuous on $[0,T]\times S_d$. In fact, the construction of φ^n implies that it is uniformly bounded with respect to the standard norm on $\mathcal{C}^1([0,T]\times S_d)$.

3.3 Uniform Convergence of DGM Approximators

We now discuss the convergence of a sequence of neural network approximators φ^n to V, the unique classical solution of the HJB equation in Equation (1.1). With the ultimate goal of establishing *uniform* convergence of the neural network approximators φ^n to the value function V, we rely on the theory of viscosity solutions to first-order, nonlinear PDEs. This powerful theory, developed by Crandall, Evans, and Lions in the 1980s for the explicit purpose of approaching HJB equations (which often lack classical, differentiable solutions) [33], will allow us to relate the neural network approximators $\varphi^n(t,m)$ to the value function V via a sequence of first-order nonlinear PDEs. Then, using a version of the comparison principle for viscosity solutions, we obtain the desired convergence.

Below, it will be useful to instead consider the HJB equation for $\eta \in \widehat{S}_d$, where

$$\widehat{S}_d := \left\{(\eta_1,\ldots,\eta_{d-1}) \in \mathbb{R}^{d-1} : \eta_j \geq 0 \text{ for all } j = 1,\ldots,d-1,\ \sum_{j=1}^{d-1}\eta_j \leq 1\right\}.$$

Then, the simplex S_d can be expressed as

$$S_d = \left\{(\eta,\eta^{-d}) : \eta \in \widehat{S}_d,\ \eta^{-d} = 1 - \sum_{j=1}^{d-1}\eta_j\right\}.$$

In turn, any function $v \in \mathcal{C}^1(S_d)$ induces a function $\widehat{v} \in \mathcal{C}^1(\widehat{S}_d)$, given by $\widehat{v}(\eta) = v(\eta,\eta^{-d})$. Furthermore, the gradient $\nabla_\eta \widehat{v}(\eta) = (\partial_{\eta_j}\widehat{v}(\eta))_{j=1}^{d-1}$ satisfies $\partial_{\eta_j}\widehat{v}(\eta) = \partial_{m_j - m_d} v(m)$. Following (3.1), we define an operator for functions defined on $[0,T] \times \widehat{S}_d$ by

$$\widehat{\mathcal{L}}[\widehat{\phi}](t,\eta) := -\partial_t \widehat{\phi}(t,\eta) + \sum_{i \in [\![d-1]\!]} \eta_i \widehat{H}^i(t,\eta,\nabla_\eta \widehat{\phi}(t,\eta)) + \eta^{-d} \widehat{H}^d(t,\eta,\nabla_\eta \widehat{\phi}(t,\eta)). \tag{3.17}$$

Now, the solution to Equation (1.1) can be written $V(t,\eta,\eta^{-d}) = \widehat{V}(t,\eta)$, where $\widehat{V} \in \mathcal{C}^{1,1}(\widehat{S}_d)$ is the unique solution to the modified HJB equation

$$\begin{aligned} &\widehat{\mathcal{L}}[\widehat{V}](t,\eta) = 0, \\ &\widehat{V}(T,\eta) = \sum_{i \in [\![d-1]\!]} \eta_i g^i(\eta,\eta^{-d}) + \eta^{-d} g^d(\eta,\eta^{-d}). \end{aligned} \tag{3.18}$$

Above, the modified Hamiltonians take inputs in $[0,T] \times \widehat{S}_d \times \mathbb{R}^{d-1}$ and are defined by

$$\begin{aligned} \widehat{H}^i(t,\eta,p) &:= H^i(t,\eta,\eta^{-d},p_1 - p_i,\ldots,p_{d-1} - p_i,-p_i), \\ \widehat{H}^d(t,\eta,p) &:= H^d(t,\eta,\eta^{-d},p,0), \end{aligned}$$

for $i \in [\![d-1]\!]$. As shown in [2], Equation (3.18) has a unique solution $\widehat{V} \in \mathcal{C}^{1,1}([0,T] \times \widehat{S}_d)$. Additionally, $\widehat{S}_d$ is a compact subset of $\mathbb{R}^{d-1}$, allowing us to apply the universal approximation theorem exactly as above, now on $\widehat{S}_d$. By the definitions of the modified Hamiltonians (in terms of the original Hamiltonians), we can also apply the exact same argument as above to obtain a result equivalent to our Theorem 3.2.4 on $\widehat{S}_d$. In particular, observe that for all $i \in [\![d-1]\!]$ and $p,p' \in \widehat{S}_d$, we have that

$$\begin{aligned} |\widehat{H}^i(t,\eta,p) - \widehat{H}^i(t,\eta,p')|^2 &= |H^i(t,\eta,\eta^{-d},p_1 - p_i,\ldots,p_{d-1} - p_i,-p_i) \\ &\qquad - H^i(t,\eta,\eta^{-d},p'_1 - p'_i,\ldots,p'_{d-1} - p'_i,-p'_i)|^2 \\ &\le C^2 \left(\sum_{j=1}^{d-1} ((p_j - p_i) - (p'_j - p'_i))^2 + (p_i - p'_i)^2 \right) \\ &\le C^2 \left(2\sum_{j=1}^{d-1} (p_j - p'_j)^2 + (2(d-1)+1)(p_i - p'_i)^2 \right) \\ &\le C^2 (2(d-1)+1) \left(\sum_{j=1}^{d-1} (p_j - p'_j)^2 + (p_i - p'_i)^2 \right) \\ &= D^2 |p - p'|^2, \end{aligned}$$

where $D^2 = 2C^2(2(d-1)+1)$. This shows that $\widehat{H}^i$ is Lipschitz continuous in p with Lipschitz constant $D \geq C > 0$ for $i \in [\![d-1]\!]$, and we similarly observe that

$$\begin{aligned}|\widehat{H}^d(t,\eta,p) - \widehat{H}^d(t,\eta,p')| &= |H^d(t,\eta,\eta^{-d},p,0) - H^d(t,\eta,\eta^{-d},p',0)| \\ &\leq D|p-p'|\end{aligned}$$

so that $\widehat{H}^i$ is Lipschitz continuous in p, with common Lipschitz constant $D > 0$, for all $i \in [\![d]\!]$. From this, the proof of a modified version of Theorem 3.2.4, now on $[0,T] \times \widehat{S}_d$, can proceed exactly as before. Note that the original value function defined on the simplex can be recovered via $V(t,\eta,\eta^{-d}) = \widehat{V}(t,\eta)$ for $\eta \in \widehat{S}_d$.

Now, we can reframe the problem in terms of Equation (3.18), which possesses a unique classical solution $\widehat{V} \in \mathcal{C}^{1,1}([0,T] \times \widehat{S}_d)$. Recall that $\widehat{S}_d \subset \mathbb{R}^{d-1}$ is the preimage of the simplex in $\mathbb{R}^d$ under the chart introduced in Equation (3.18). Working with Equation (3.18) rather than Equation (1.1) allows us to cite results from the theory of viscosity solutions that require the domain of the relevant PDE to be open; note that $\mathrm{Int}(\widehat{S}_d)$ is an open subset of $\mathbb{R}^{d-1}$ whereas S_d has empty interior in $\mathbb{R}^d$. Additionally, the following result demonstrates that the convergence result on $\widehat{S}_d$ translates to S_d without any issues.

Proposition 3.3.1. *Assume that $\widehat{V} \in \mathcal{C}([0,T] \times \widehat{S}_d)$ and $\widehat{\phi}^n \in \mathcal{C}([0,T] \times \widehat{S}_d)$ are such that $\|\widehat{V} - \widehat{\phi}^n\|_\infty \to 0$ as $n \to \infty$. Then, $\|V - \phi^n\|_\infty \to 0$ as $n \to \infty$, where $V, \phi^n \in \mathcal{C}([0,T] \times S_d)$ are given by $V(t,\eta,\eta^{-d}) = \widehat{V}(t,\eta)$ and $\phi^n(t,\eta,\eta^{-d}) = \widehat{\phi}^n(t,\eta)$ for all $(t,\eta,\eta^{-d}) \in [0,T] \times S_d$ and all $n \in \mathbb{N}$.*

We remark that the opposite direction is also true; given a sequence of functions on $[0,T] \times S_d$ that converge uniformly to $V \in \mathcal{C}([0,T] \times S_d)$, the corresponding functions on $[0,T] \times \widehat{S}_d$ converge to $\widehat{V} \in \mathcal{C}([0,T] \times \widehat{S}_d)$, given by $\widehat{V}(t,\eta) = V(t,\eta,\eta^{-d})$ in our notation above. However, we do not require the converse of Proposition 3.3.1, and the proof is analogous.

Proof. This is a simple consequence of the definition of $\widehat{S}_d$. Indeed, if $\|\widehat{V} - \widehat{\phi}^n\|_\infty \to 0$ as $n \to \infty$. Then, for any $\varepsilon > 0$, we have that for all $n \in \mathbb{N}$ sufficiently large,

$$\sup_{(t,\eta) \in [0,T] \times \widehat{S}_d} |\widehat{V}(t,\eta) - \widehat{\phi}^n(t,\eta)| < \varepsilon.$$

Consequently, for all $n \in \mathbb{N}$ sufficiently large, we have that

$$\begin{aligned}\sup_{(t,\eta,\eta^{-d})\in[0,T]\times S_d} |V(t,\eta,\eta^{-d}) - \phi^n(t,\eta,\eta^{-d})| &= \sup_{(t,\eta)\in[0,T]\times \widehat{S}_d} |\widehat{V}(t,\eta) - \widehat{\phi}^n(t,\eta)| \\ &= \sup_{(t,\eta)\in[0,T]\times \widehat{S}_d} |\widehat{V}(t,\eta) - \widehat{\phi}^n(t,\eta)| \\ &< \varepsilon.\end{aligned}$$

This means precisely that $\|V - \phi^n\|_\infty \to 0$ as $n \to \infty$. □

As a consequence of the above proposition, it suffices to show the uniform convergence of a sequence of neural network approximators $\widehat{\varphi}^n$ to the unique classical solution $\widehat{V}$ of Equation (3.18) on $\widehat{S}_d$, as we can then recover uniform convergence on the simplex. Note that from now on, we consider neural networks, denoted by $\widehat{\varphi}$, which are defined as in (3.12) but with the input $\eta \in \widehat{S}_d$ rather than $m \in S_d$. From the discussion preceding Equation (3.18), we also know that Theorem 3.2.4 holds on $\widehat{S}_d$, yielding the existence of a sequence of neural networks $\{\widehat{\varphi}^n(t,\eta) := \widehat{\varphi}(t,\eta;\theta^n)\}_{n\in\mathbb{N}}$ such that $L(\theta^n) \to 0$ as $n \to \infty$. In turn, each network $\widehat{\varphi}^n(t,\eta)$ satisfies its own "perturbed" PDE, of the form

$$\widehat{\mathcal{L}}[\widehat{\varphi}^n](t,\eta) = \widehat{e^n}(t,\eta), \qquad (t,\eta) \in [0,T] \times \widehat{S}_d, \tag{3.19}$$

with $\widehat{e^n}(t,\eta) := e^n(t,\eta,\eta^{-d})$. For notational simplicity, we take

$$\widehat{G}(\eta) := \sum_{i\in[\![d-1]\!]} \eta_i g^i(\eta,\eta^{-d}) + \eta^{-d} g^d(\eta,\eta^{-d})$$

in this section to denote the terminal condition of the HJB equation on $\widehat{S}_d$. Denoting $\widehat{\Omega}_T := [0,T] \times \widehat{S}_d$ as in the previous section, Theorem 3.2.4 above implies that Equation (3.19) satisfies

$$\max_{(t,\eta)\in[0,T]\times\widehat{S}_d} \left|\widehat{e^n}(t,\eta)\right| + \max_{\eta\in\widehat{S}_d} \left|\widehat{\varphi}^n(T,\eta) - \widehat{G}(\eta)\right| \to 0 \qquad \text{as} \qquad n \to \infty.$$

With this context in mind, we state the main *convergence result* of this section.

Theorem 3.3.2. *The family of neural network approximators* $\{\widehat{\varphi}^n(t,\eta)\}_{n\in\mathbb{N}}$ *satisfying Equation* (3.19) *converges uniformly to* $\widehat{V} \in \mathcal{C}^{1,1}([0,T]\times\widehat{S}_d)$*, the unique classical solution* *of Equation* (3.18)*, in the sense that*

$$\sup_{(t,\eta)\in[0,T]\times\widehat{S}_d} |\widehat{\varphi}^n(t,\eta) - \widehat{V}(t,\eta)| \to 0 \qquad as \qquad n \to \infty.$$

To prove the above theorem, we argue via the comparison principle for viscosity solutions to (1.1) presented in [2]. To this end, we require a suitable definition of viscosity solutions of Equation (3.18) on $\widehat{S}_d$.

Definition 3.3.3. *A function $\widehat{v} \in \mathcal{C}((0,T) \times Int(\widehat{S}_d))$ is:*

(i) a viscosity subsolution of Equation (3.18) *if for any $\widehat{\varphi} \in \mathcal{C}^1((0,T)\times Int(\widehat{S}_d))$, $\widehat{\mathcal{L}}[\widehat{\varphi}] \leq 0$ for every local maximum $(t_0, \eta_0) \in (0,T) \times Int(\widehat{S}_d)$ of $\widehat{v} - \widehat{\varphi}$ on $(0,T) \times Int(\widehat{S}_d)$.*

(ii) a viscosity supersolution of Equation (3.18) *if for any $\widehat{\varphi} \in \mathcal{C}^1((0,T) \times Int(\widehat{S}_d))$, $\widehat{\mathcal{L}}[\widehat{\varphi}] \geq 0$ for every local minimum $(t_0, \eta_0) \in (0,T) \times Int(\widehat{S}_d)$ of $\widehat{v} - \widehat{\varphi}$ on $(0,T) \times Int(\widehat{S}_d)$.*

(iii) a viscosity solution of Equation (3.18) *if $\widehat{v}$ is both a viscosity subsolution and viscosity supersolution.*

Remark 3.3.4. When viscosity solutions are introduced in [2], the author also allows for test functions on $[0,T) \times S_d$ (resp. $[0,T) \times \widehat{S}_d$), noting that $[0,T) \times S_d$ (resp. $[0,T) \times \widehat{S}_d$) is no longer an open subdomain of $\mathbb{R}^{d+1}$ (resp. $\mathbb{R}^d$). However, in order to utilize [33, Theorem 3.3], the standard comparison principle for viscosity solutions, we must consider viscosity solutions on open subdomains of $\mathbb{R}^d$. As noted in [2], it is also not immediately clear that a classical solution to Equation (1.1) is a viscosity solution if the latter is defined on a closed set.

We could alternatively cite the comparison principle from [2, Theorem 3.4] that utilizes the definition of viscosity solutions on closed sets presented therein. However, in order to utilize the clearly presented stability properties of viscosity solutions under uniform limits presented in [32, 33], we opt for the standard definition in Definition 3.3.3.

In order to establish Theorem 3.3.2, we proceed using a standard comparison principle argument for viscosity solutions that also leverages the fact that $\widehat{V} \in \mathcal{C}^{1,1}([0,T] \times \widehat{S}_d)$ is the unique viscosity solution to Equation (3.18) from [2].

Proof of Theorem 3.3.2. For each $n \in \mathbb{N}$, we may define an operator

$$\widehat{\mathcal{L}}^n[\widehat{\phi}](t,\eta) := -\partial_t\widehat{\phi}(t,\eta) + \sum_{i\in[\![d-1]\!]} \eta_i \widehat{H}^i(t,\eta,\nabla_\eta\widehat{\phi}(t,\eta)) + \eta^{-d}\widehat{H}^d(t,\eta,\nabla_\eta\widehat{\phi}(t,\eta)) - \widehat{e^n}(t,\eta),$$

corresponding to the sequence of PDEs described in (3.19). Because $\widehat{\mathcal{L}}^n[\widehat{\phi}]$ depends only on the derivatives of $\widehat{\phi}$ (and not on $\widehat{\phi}$ itself), we observe that $\widehat{\mathcal{L}}^n$ is *proper* in the sense of [33]. This fact also ensures that the technical conditions preceding the comparison principle [33, Theorem 3.3] are satisfied.

Now, note that by the discussion following Proposition 3.3.1,

$$\max_{(t,\eta)\in[0,T]\times\widehat{S}_d} \left|\widehat{e^n}(t,\eta)\right| \to 0 \qquad \text{as} \qquad n\to\infty,$$

meaning that $\widehat{e^n}$ converges uniformly to zero on $[0,T]\times\widehat{S}_d$. Now, for each $n\in\mathbb{N}$, define $T^n : [0,T]\times\widehat{S}_d\times\mathbb{R}\times\mathbb{R}^{d-1}\to\mathbb{R}$ by

$$T^n(t,\eta,p_0,p) := -p_0 + \sum_{i\in[\![d-1]\!]} \eta_i\widehat{H}^i(t,\eta,p) + p^{-d}\widehat{H}^d(t,\eta,p) - \widehat{e^n}(t,\eta).$$

We then have that T^n converges uniformly on $[0,T]\times\widehat{S}_d\times\mathbb{R}\times\mathbb{R}^{d-1}$ to

$$T(t,\eta,p_0,p) := -p_0 + \sum_{i\in[\![d-1]\!]} \eta_i\widehat{H}^i(t,\eta,p) + p^{-d}\widehat{H}^d(t,\eta,p).$$

These definitions are motivated by the fact that Equation (1.1) can be written succinctly as

$$T(t,\eta,\partial_t\widehat{\varphi},\nabla_\eta\widehat{\varphi}) = 0,$$

while Equation (3.19) is given by

$$T^n(t,\eta,\partial_t\widehat{\varphi},\nabla_\eta\widehat{\varphi}) = 0$$

for each $n\in\mathbb{N}$. Now, following [33, Remark 6.3], we note that because $\widehat{\varphi}^n$ is a classical solution (and therefore a viscosity solution) to the equation $T^n(t,\eta,\partial_t\widehat{\varphi},\nabla_\eta\widehat{\varphi}) = 0$ on

$(0,T) \times \mathrm{Int}(\widehat{S}_d)$, then

$$\overline{V}(t,\eta) := \lim_{j\to\infty} \sup\{\widehat{\varphi}^n(s,\nu) : n \geq j,\ (t,\eta) \in [0,T) \times \widehat{S}_d,\ |(s,\nu)-(t,\eta)| \leq 1/j\}$$

is a viscosity subsolution to the equation $T(t,\eta,\partial_t\widehat{\varphi},\nabla_\eta\widehat{\varphi}) = 0$, as we have that

$$T(t,\eta,p_0,p) = \liminf_{n\to\infty} T^n(t,\eta,p,p_0).$$

On the other hand, we also observe that

$$\underline{V}(t,\eta) := \lim_{j\to\infty} \inf\{\widehat{\varphi}^n(s,\nu) : n \geq j,\ (t,\eta) \in [0,T) \times \widehat{S}_d,\ |(s,\nu)-(t,\eta)| \leq 1/j\}$$

is a viscosity supersolution to the equation $T(t,\eta,\partial_t\widehat{\varphi},\nabla_\eta\widehat{\varphi}) = 0$ by the same reasoning. By construction, observe that $\underline{V} \leq \overline{V}$ on $[0,T) \times \widehat{S}_d$. Note also that both $\underline{V}$ and $\overline{V}$ are well-defined on $\{0\} \times \partial\widehat{S}_d$ by their construction. However, by the comparison principle presented in [33, Theorem 3.3], the fact that $\underline{V}$ is a viscosity supersolution and $\overline{V}$ is a viscosity subsolution is sufficient to conclude that $\overline{V} \leq \underline{V}$ on $[0,T) \times \widehat{S}_d$, observing that the comparison principle still holds on the closure of the domain $(0,T) \times \mathrm{Int}(\widehat{S}_d)$.

In particular, $\overline{V} = \underline{V}$ is a viscosity solution. As shown in [2, Theorem 9], Equation (1.1) has a unique viscosity solution $\widehat{V}$, showing that $\overline{V} = \underline{V} = \widehat{V}$. Now, [33, Remark 6.4] implies that $\lim_{n\to\infty} \widehat{\varphi}^n(t,\eta) = \widehat{V}(t,\eta)$ uniformly on $[0,T) \times \widehat{S}_d$.

Finally, note that we also have that

$$\max_{\eta\in\widehat{S}_d} \left|\widehat{\varphi}^n(T,\eta) - \widehat{V}(T,\eta)\right| \to 0,$$

from the construction of the modified DGM loss, allowing us to conclude uniform convergence of $\widehat{\varphi}^n \to \widehat{V}$ on the entire region $[0,T] \times \widehat{S}_d$ as claimed. □

3.4 Comparison to DGM Algorithm with L^2-Loss

At this point, we can clarify the reasons for the modification to the DGM algorithm made in Section 3.1. Here, we refer to the Sobolev space $W^{k,p}(\Omega)$, given by the space of functions $f \in L^p(\Omega)$ such that for any multi-index α with $|\alpha| \leq k$, the derivative $D^\alpha f$ exists and belongs to $L^p(\Omega)$ itself; see [55, Chapter 5] for more details on this characterization of Sobolev spaces. We also define the space $L^p(0,T;W^{k,q}(\Omega))$ by the function $f \in L^p([0,T]\times\Omega)$ such that, for fixed $t\in[0,T]$, $f(t,\cdot)\in W^{k,q}(\Omega)$. The spaces $L^p(0,T;L^q(\Omega))$ are then defined analogously for $1\le p,q\le\infty$. The authors of [1] formulated the L^2-loss,[1] as

$$\tilde{L}(\theta) = \|\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t,m)\|^2_{2,[0,T]\times S_d,\nu_1} + \|\varphi(T,m;\theta) - \sum_{i\in[\![d]\!]} m_i g^i(m)\|^2_{2,S_d,\nu_2}, \tag{3.20}$$

because of the natural connection between the class of equations that they considered and convergence in L^2. A key step in the proof of their analog to Theorem 3.3.2 involves obtaining a uniform bound on $\{\varphi(t,m;\theta^n)\}_{n\in\mathbb{N}}$ in $L^\infty(0,T;L^2(\Omega))\cap L^2(0,T;W^{1,2}(\Omega))$, where Ω is the open domain on which the PDE is considered. In turn, this arises from an energy bound on quasilinear parabolic equations such as the one presented in [56], or in more generality in [57]. However, such a bound only holds for equations of the form

$$\begin{cases} \partial_t u - \operatorname{div}(a(t,x,u,\nabla u)) = H(t,x,\nabla u) & (t,x)\in(0,T)\times\Omega, \\ u = 0 & (t,x)\in(0,T)\times\partial\Omega, \\ u(0,x) = u_0(x) & x\in\Omega. \end{cases}$$

that satisfy the Leray–Lions conditions. Namely, there must exist $\alpha>0$ such that

$$\alpha|\xi|^p \le a(t,x,p,\xi)\cdot\xi$$

for all $\xi\in\mathbb{R}^d$ and some $1<p<d$. Clearly, this fails in our case, where a is identically zero, even though our HJB equation otherwise satisfies the structure conditions in [57] Following the discussion in Section 3.2, it may be possible to obtain a similar uniform bound by bounding the networks and their weights without losing any universal approximation guarantees, but we do not currently consider this approach.

Translating the convergence argument in [1, Theorem 7.3] to our context is also complicated by the fact that the class of quasilinear parabolic PDEs for which they prove convergence of the DGM possesses a standard notion of weak solutions that, via the dominated convergence theorem, cooperates with convergence in L^2. In the case of HJB

[1]As with the DGM loss in Equation (3.2), this loss is computed in practice by sampling points according to probability measures ν_1 and ν_2 on $[0,T]\times S_d$ and S_d respectively to obtain an unbiased estimate of the L^2-loss in Equation (3.20).

equations, however, viscosity solutions take the place of weak solutions and instead behave nicely with respect to uniform convergence. Thus, we require that $\widehat{e^n} \to 0$ uniformly on $[0,T] \times \widehat{S}_d$ in Equation (3.19), but the formulation of the DGM with L^2-loss only implies convergence of the error term in L^2.

Finally, [1] only concludes uniform convergence of the neural network approximators to the true solution of the PDE after imposing additional assumptions of uniform boundedness and equicontinuity on the neural networks. Although this may be a reasonable assumption to include, we find that our modified DGM algorithm and the theory of viscosity solutions provide a more direct route to uniform convergence; see Section 3.4 for a more detailed discussion of the equicontinuity of neural network approximators and the potential issues with such an approach.

As discussed above, the proof technique via a comparison principle presented in Theorem 3.3.2 no longer holds for the DGM with L^2-loss, given by Equation (3.20). However, we *do* still have an analogue of Theorem 3.2.4 for the loss functional defined in Equation (3.20). For convenience, we recall that the DGM algorithm with L^2-loss aims to minimize the L^2-error of the approximate solution to the PDE in question. Specifically, as presented in [1], DGM learns an approximator $\varphi(t,m;\theta)$, parametrized by θ, by minimizing the L^2-loss of the HJB equation:

$$\tilde{L}(\theta) := \|\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t,m)\|^2_{2,[0,T]\times S_d,\nu_1} + \|\varphi(T,m;\theta) - \sum_{i\in[\![d]\!]} m_i g^i(m)\|^2_{2,S_d,\nu_2}.$$

Above, ν_1 and ν_2 are probability densities on $[0,T]\times S_d$ and S_d respectively. We then have the following analog to Theorem 3.2.4:

Theorem 3.4.1. *Let $\sigma \in \mathcal{C}^1(\mathbb{R})$ be bounded and nonconstant. For every $\varepsilon > 0$, there exists a constant $K(d,T,C) > 0$, where d is the dimension of the simplex S_d, T is the finite time horizon of the MFCP, and C is the Lipschitz constant of the PDE-Hamiltonian in Equation* (1.1)*, such that for some $\varphi = \varphi(\cdot,\cdot;\theta) \in \mathfrak{C}_{d+1}(\sigma)$, the DGM loss functional in Equation* (3.20) *satisfies $\tilde{L}(\theta) \le K\varepsilon$.*

Proof. The proof of this theorem is similar to that of Theorem 3.2.4, with a few slight modifications. In particular, the Liptschitz continuity of the PDE-Hamiltonian yields

$$\begin{aligned}
&\int_{\Omega_T} \left| \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^i\varphi(t,m;\theta)) - \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^iV(t,m)) \right|^2 d\nu_1(t,m) \\
&= \int_{\Omega_T} \left| \sum_{i\in[\![d]\!]} m_i (H^i(t,m,D^i\varphi(t,m;\theta)) - H^i(t,m,D^iV(t,m))) \right|^2 d\nu_1(t,m) \\
&\leq d \sum_{i\in[\![d]\!]} \int_{\Omega_T} \left| m_i (H^i(t,m,D^i\varphi(t,m;\theta)) - H^i(t,m,D^iV(t,m))) \right|^2 d\nu_1(t,m) \\
&\leq dC^2 \sum_{i\in[\![d]\!]} \int_{\Omega_T} |D^i\varphi(t,m;\theta) - D^iV(t,m)|^2 d\nu_1(t,m).
\end{aligned} \tag{3.21}$$

Above, we apply the Cauchy–Schwarz inequality in the second-to-last line above and note that $|m_i| \leq 1$ for any $m \in S_d$. Now, for each $i \in [\![d]\!]$ and any $(t,m) \in \Omega_T$, observe that by denoting the standard basis of $\mathbb{R}^d$ by $\{e_i\}_{i\in[\![d]\!]}$, we have that

$$\begin{aligned}
|D^i\varphi(t,m;\theta) - D^iV(t,m)|^2 &= \frac{1}{2}\sum_{j=1}^d |(\nabla_m\varphi(t,m;\theta) - \nabla_m V(t,m)) \cdot (e_j - e_i)|^2 \\
&\leq \frac{1}{2}\sum_{j=1}^d |\nabla_m V(t,m) - \nabla_m\varphi(t,m;\theta)|^2 |e_j - e_i|^2 \\
&\leq d|\nabla_m V(t,m) - \nabla_m\varphi(t,m;\theta)|^2,
\end{aligned}$$

again by applying the Cauchy–Schwarz inequality. Reusing the inequality from (3.14), we can bound (3.21) by

$$\begin{aligned}
&dC^2 \sum_{i\in[\![d]\!]} \int_{\Omega_T} |D^i\varphi(t,m;\theta) - D^iV(t,m)|^2 d\nu_1(t,m) \\
&\leq d^3C^2 \int_{\Omega_T} |\nabla_m (V(t,m) - \varphi(t,m;\theta))|^2 d\nu_1(t,m) \\
&\leq K\varepsilon^2
\end{aligned}$$

for some positive constant $K = K(d,T,C) > 0$ by the construction of φ. Finally, because the value function V satisfies $\mathcal{L}[V](t,m) = 0$ for all $(t,m) \in \Omega_T$, we may write

$$\begin{aligned}
\tilde{L}(\theta) &= \|\mathcal{L}[\varphi](t,m)\|^2_{2,\Omega_T,\nu_1} + \|\varphi(T,m;\theta) - \sum_{i\in[\![d]\!]} m_i g^i(m)\|^2_{2,S_d,\nu_2} \\
&= \|\mathcal{L}[\varphi](t,m) \ - \mathcal{L}[V](t,m)\|^2_{2,\Omega_T,\nu_1} + \|\varphi(T,m;\theta) - V(T,m)\|^2_{2,S_d,\nu_2} \\
&\leq 2\int_{\Omega_T} \left| \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^i\varphi(t,m;\theta)) - \sum_{i\in[\![d]\!]} m_i H^i(t,m,D^iV(t,m)) \right|^2 d\nu_1(t,m) \\
&\quad + 2\int_{\Omega_T} |\partial_t V(t,m) - \partial_t\varphi(t,m;\theta)|^2 \, d\nu_1(t,m) \\
&\quad + \int_{S_d} |\varphi(T,m;\theta) - V(T,m)|^2 d\nu_2(m) \\
&\leq K\varepsilon^2
\end{aligned}$$

by applying the Cauchy–Schwarz inequality yet again, taking K larger if necessary, and noting that the estimate in (3.13) provides bounds on the two remaining terms in the above expression. □

Remark 3.4.2. In the case of the DGM algorithm with L^2-loss, the measures ν_1 and ν_2, regardless of the densities that they correspond to, are defined as probability measures on $[0,T] \times S_d$ and S_d respectively. Thus, the above result is *independent* of the choice of densities ν_1 and ν_2, as we simply use the bounds

$$\int_{\Omega_T} |\partial_t V(t,m) - \partial_t\varphi(t,m;\theta)|^2 \, d\nu_1(t,m) \leq \varepsilon^2 \nu_1(\Omega_T) = \varepsilon^2$$

and

$$\int_{S_d} |\varphi(T,m;\theta) - V(T,m)|^2 d\nu_2(m) \leq \varepsilon^2 \nu_2(S_d) = \varepsilon^2$$

respectively.

3.4.1 Equicontinuous and Uniformly-Bounded Neural Networks

Recall that in [1, Theorem 7.3], the authors require additional assumptions of both equicontinuity and uniform boundedness of neural network approximators that we bypass via the theory of viscosity solutions. The primary condition that allows for equicontinuity is the boundedness of the weights in the hidden layer(s) of the neural network used to approximate some continuous function. Given the discussion in [48], which analyzes the approximation power of networks with bounded weights, it is likely possible to approximate $V \in \mathcal{C}^{1,1}([0,T] \times S_d)$ (in the standard norm on $\mathcal{C}^1$, as in Proposition 3.2.1) via a sequence of neural networks with bounded weights. If this is the case, we may apply the equicontinuity results established in [58], obtaining an *equicontinuous* sequence of neural network approximators to the value function V. Finally, the argument of [1] provides for uniform boundedness, allowing us to apply the Arzelà–Ascoli theorem to obtain uniform convergence.

This discussion is summarized more precisely below in a partial result. Importantly, the following result only holds for two-layer neural networks. To this end, we consider two layer networks of the form

$$\mathfrak{C}^{(n)}_{2,d+1}(\sigma) := \left\{ \varphi : \mathbb{R}^{d+1} \to \mathbb{R} \;\middle|\; \varphi(t,x;\theta) = \sum_{i=1}^{n} \beta_i \sigma\left(\alpha_{1,i} t + \sum_{j=1}^{d} \alpha_{j+1,i} x_j + c_i \right),\ \theta \in \mathbb{R}^{2n+n(d+1)} \right\}.$$

Each network in this class has weights given by $\theta = (\beta_1, \ldots, \beta_n, \alpha_{1,1}, \ldots, \alpha_{d+1,n}, c_1, \ldots, c_n) \in \mathbb{R}^{2n+n(d+1)}$. We then denote

$$\mathfrak{C}_{2,d+1}(\sigma) := \bigcup_{n=1}^{\infty} \mathfrak{C}^{(n)}_{2,d+1}(\sigma).$$

With this notation out of the way, we can state the following theorem.

Theorem 3.4.3. *Take $\mathfrak{C}_{2,d+1}(\sigma)$ as defined above and consider any function $f \in \mathcal{C}^m(K)$, for a compact set $K \subset \mathbb{R}^{d+1}$. Let $M > 0$ be such that $\sup_{x\in K} |f(x)| \le M$. Denote by $\mathfrak{C}'_{2,d+1}(\sigma)$ the subset of networks with weights $\theta = (\beta_1, \ldots, \beta_n, \alpha_{1,1}, \ldots, \alpha_{d+1,n}, c_1, \ldots, c_n) \in \mathbb{R}^{2n+n(d+1)}$ satisfying $|\alpha_{j,i}| \le M$, $|c_i| \le M$, and $|\beta_i| \le M$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, d+1$. Then, for any $m \in \mathbb{N}$, there exists a sequence of neural networks $\{\varphi^k(t,m) := \varphi(t,m;\theta^k)\}_{k\in\mathbb{N}}$ with an increasing number of hidden units such that $\|f - \varphi^k\|_{\mathcal{C}^m(K)} < 1/k$ for all $k \in \mathbb{N}$. Furthermore, the sequence $\{\varphi^k\}_{k\in\mathbb{N}}$ is equicontinuous with respect to the inputs $(t,m) \in [0,T] \times S_d$.*

Proof. As shown in [48], any continuous, bounded functions can be uniformly approximated by neural networks with bounded weights. Assuming that we can obtain a universal approximation result for neural networks in $\mathfrak{C}'_{2,d+1}$, [58, Proposition 8] shows that the weights of any network $\varphi(\cdot,\cdot;\theta) \in \mathfrak{C}'_{2,d+1}(\sigma)$ with n hidden units such that

$\|f - \varphi(\cdot, \cdot; \theta)\|_\infty < 1$ must have weights satisfying

$$\left| \sum_{i=1}^{n} \beta_i \right| \leq M + 1.$$

In turn, the above condition allows us to apply [58, Theorem 20], which states that the subset of $\mathfrak{C}'_{2,d+1}(\sigma)$ satisfying the above summability condition is equicontinuous with respect to the input space. Specifically, this implies that the sequence $\{\varphi^k\}_{k\in\mathbb{N}}$ is equicontinuous with respect to the inputs $(t, m) \in [0, T] \times S_d$ as claimed. □

In order to utilize Theorem 3.4.3 in our context, however, we require a suitable version of the universal approximation theorem for neural networks with bounded weights. Despite a thorough literature review, it does not seem that the universal approximation result in Proposition 3.2.1 has an analogue for networks with uniformly bounded weights. In particular, there is no current result that allows for the approximation of an arbitrary function $h \in \mathcal{C}^m(\mathbb{R}^{d+1})$ *and* its derivatives on a compact set using networks with bounded weights. For the time being, the DGM algorithm with uniform loss and Theorem 3.3.2 provide a workaround for this issue.

3.4.2 Stability Properties of the DGM

Here, we also note an interesting connection to recent work in the field of physics-informed neural networks (PINNs), a recent framework for incorporating PDE constraints into the training of neural networks. Specifically [59] show that, for a class of second-order HJB equations arising from stochastic control problems, it is *impossible* to obtain a convergence guarantee along the lines of Theorem 3.3.2 if L^2-loss is used. We emphasize that this result does not necessarily apply to the first-order HJB equation arising from the MFCP. Our work is further complicated by the fact that Equation (1.1) has solutions on the simplex $[0, T] \times S_d$, whereas traditional stochastic control problems (for which the HJB equation is derived via a dynamic programming principle) are solved over $[0, T] \times \mathbb{R}^n$. However, future work may investigate whether the stability results obtained in [59] generalize to the context of the MFCP. Currently, this connection only provides some intuition as to why the L^∞-loss may be the appropriate choice for training neural networks to solve Equation (1.1).

To make the above statement slightly more precise, we define an operator for second-order HJB equations on $\mathbb{R}^n \times [0,T]$, given by

$$\mathcal{T}[u](x,t) := \partial_t u(x,t) + \frac{1}{2}\sigma^2 \Delta u(x,t) + \min_{m \in \mathcal{M}} \{ r(x, m(t,x), t) + \nabla u \cdot m_t.$$

In turn, [59] considers HJB equations of the form

$$\begin{cases} \mathcal{T}[u](x,t) = 0 & x \in \mathbb{R}^n \times [0,T] \\ u(x,T) = g(x), \end{cases} \tag{3.22}$$

where $\mathcal{M}$ is a set of admissible (feedback) controls and σ is a parameter that governs the evaluation of a state $\{X_t\}_{0 \le t \le T}$ of a stochastic control problem via the SDE

$$\begin{cases} dX_s = m(s, X_s)ds + \sigma dW_s, & s \in [0,T] \\ X_0 = x. \end{cases}$$

Above $\{W_s\}_{s \in [0,T]}$ is a standard Brownian motion. In [59], the authors impose standard assumptions in stochastic control to ensure that the solution to Equation (3.22) is unique [34]. As usual, the objective of the stochastic control problem is to minimized the expected cost

$$J_{x,t}(m) := \mathbb{E}\left[\int_t^T f(X_s, m, s) ds + g(X_T) \right],$$

where the expectation is taken over the randomness of the stochastic process $\{X_t\}_{0 \le t \le T}$, the rate function $f : \mathbb{R}^n \times \mathbb{R}^n \times [0,T] \to \mathbb{R}$ denotes the running cost of the control problem, and $g : \mathbb{R}^n \to \mathbb{R}$ denotes the terminal cost. The authors in [59] also assume that $f(x,m) = a_1|m_1|^{\alpha_1} + \ldots + a_n|m_n|^{\alpha_n} - \varphi(x,t)$ for coefficients $a_1, \ldots, a_n, \alpha_1, \ldots, \alpha_n \in \mathbb{R}$ and some function $\varphi \in \mathcal{C}(\mathbb{R}^n \times [0,T])$. Under this assumption, one can write the first-order term in Equation (3.22) as

$$\min_{m \in \mathcal{M}} \{ r(x, m(t,x), t) + \nabla u \cdot m_t \} = -\sum_{i=1}^n A(a_i, \alpha_i) |\partial_{x_i} u|^{c_i(a_i, \alpha_i)} - \varphi(x,t).$$

Although somewhat restrictive, this assumption applies to common settings in stochastic control such as linear–quadratic–Gaussian control. Before stating the main result of [59] and its relevance to our work, we first require a definition.

Definition 3.4.4. *Let $(V_1, \|\cdot\|_1), (V_2, \|\cdot\|_2), (V_3, \|\cdot\|_3)$ be Banach spaces such that $u \in V_1 \cap V_3$ and $u(T, \cdot) \in V_2$, where u is the unique solution to Equation* (3.22). *Then, Equation* (3.22) *is (V_1, V_2, V_3)-stable if for any $v \in V_1 \cap V_3$ with $v(T, \cdot) \in V_2$,*

$$\|u - v\|_3 \leq C \left(\|\mathcal{T} v\|_1 + \|v(T, \cdot) - g\|_2 \right)$$

for some constant C that does not depend on v.

Intuitively, the above definition gives a condition under which training the DGM with loss given by the norms on V_1 and V_2, respectively, will yield convergence to the true solution of Equation (3.22) with respect to the norm on V_3. For instance, if one aims to apply the standard DGM algorithm with L^2-loss to solve a PDE on a region $\Omega \times [0, T] \subset \mathbb{R}^n \times [0, T]$, one could take $V_1 = L^2(\Omega \times [0, T])$, $V_2 = L^2(\Omega)$, and $V_3 = \mathcal{C}(\Omega \times [0, T])$. If such a PDE is (V_1, V_2, V_3)-stable, then training the DGM with L^2-loss will ensure uniform convergence to the true solution. Similarly, in Theorem 3.3.2, we show that with $V_1 = L^\infty([0, T] \times \widehat{S}_d)$, $V_2 = L^\infty(\widehat{S}_d)$, and $V_3 = L^\infty([0, T] \times \widehat{S}_d)$, Equation (3.18) is (V_1, V_2, V_3)-complete.

Having defined the stability of Equation (3.22) as above, we state some relevant results from [59]. Specifically, the authors therein show the following two propositions:

Proposition 3.4.5. *[59, Theorem 4.3] For $p, q \geq 1$, let $r_0 = \frac{(n+2)q}{n+q}$ and $\bar{c} = \max_{1 \leq i \leq n} c_i$, with the coefficients c_i as in Equation* (3.22). *Assume that the following conditions hold on p, q, r_0:*

$$p \geq \max\{2, (1 - 1/\bar{c})n\}, \quad q > \frac{(\bar{c} - 1)n^2}{(2 - \bar{c})n + 2}, \quad \frac{1}{r_0} \geq \frac{1}{p} - \frac{1}{n}.$$

Then, for any $r \in [1, r_0)$ and bounded open subdomain $\Omega \subset \mathbb{R}^n \times [0, T]$, Equation (3.22) *is $(L^p(\mathbb{R}^n \times [0, T]), L^q(\mathbb{R}^n), W^{1,r}(Q))$-stable for $\bar{c} \leq 2$.*

Loosely speaking above theorem implies that stability is achieved when $p, q \gg n$, a condition that is not met when applying the DGM, with L^2-loss, to high-dimensional HJB equations. Given the preceding result and the fact that if $f \in L^\infty(\mathbb{R}^n \times [0, T]) \cap L^q(\mathbb{R}^n \times [0, T])$ for all $q \geq 1$, then $\|f\|_{L^\infty} = \lim_{p \to \infty} \|f\|_{L^p}$, we might expect the DGM with L^∞-loss to be stable, precisely as we show in the case of the MFCP. Furthermore, [59] also provides the following theorem showing that for *any* $p \leq n/4$, there exists an HJB equation as in Equation (3.22) such that the L^2-loss for the DGM is arbitrarily small, yet the distance between the true solution to the HJB equation and the DGM approximation is arbitarily large. More precisely, we have the following:

Proposition 3.4.6. *[59, Theorem 4.4] There exists an equation of the form Equation* (3.22)*, with unique solution u, such that for any $\varepsilon > 0$, $C > 0$, $r \geq 1$, $m \in \mathbb{N}$, and $p \in [1, n/4]$, there exists a function $v \in \mathcal{C}^\infty(\mathbb{R}^n \times [0,T])$ such that the following two conditions hold:*

(1) $\|\mathcal{T}v - \varphi\|_{L^p(\mathbb{R}^n \times [0,T])} < \varepsilon$, $v(T, \cdot) = g$, and $u - v$ is compactly-supported.

(2) $\|u - v\|_{W^{m,r}(\mathbb{R}^n \times [0,T])} > C$.

In summary, while the DGM with respect to L^p-loss for p large (or $p = \infty$) provides stability guarantees in the sense of Definition 3.4.4, if $p \leq n/4$, then carrying out the DGM algorithm with L^p-loss provides no guarantee that the resulting neural network approximation will converge to the true solution to Equation (3.22). In particular, the standard L^2-loss approach from [1] will not suffice.

We again emphasize that the result stated above in Proposition 3.4.6 only holds for the second-order HJB equation in Equation (3.22), with domain $\mathbb{R}^n \times [0,T]$. However, the recent progress made in [59] provides, at the very least, heuristic insight into the gap between our proof technique, which relies on an L^∞-loss formulation of the DGM, and the original L^2-loss approach. Investigating such stability properties for Equation (1.1) remains the subject of future work.

Chapter 4

Numerical Experiments

4.1 DGM Experiments

In this section, we present numerical results for the DGM applied to a simple example case of the MFCP, as presented in [2, Example 2]. In particular, we work with the following example.

Example. Consider the quadratic running cost

$$f(t,i,\alpha,m) = \frac{1}{2}\sum_{j\in[\![d]\!],j\neq i} c_{i,j}\alpha_{i,j}^2 + f_0^i(m),$$

with $f_0^i(m) := m_i$ and $\{c_{i,j}\}_{i,j\in[\![d]\!]} \in \mathbb{R}^{d\times d}$ a cost matrix that encodes the cost of transitioning from state i to state j for $i \neq j$. We consider the linear terminal condition given by

$$g^i(m) = m_i$$

for $i \in [\![d]\!]$. With this choice of terminal cost, we obtain the terminal condition

$$V(T,m) = G(m) = \sum_{i\in[\![d]\!]} m_i^2.$$

In this simple example, derived in [2], the Hamiltonian is explicitly given

$$H^i(t,m,z) = \sum_{j\neq i}\left(-\mathfrak{a}^*(-z_j)z_j - \frac{1}{2}(\mathfrak{a}^*(-z_j))^2\right) - f_0^i(m),$$

where

$$\mathfrak{a}^*(s) = \begin{cases} 0 & s \le 0, \\ s & 0 \le s \le M, \\ M & s \ge M. \end{cases}$$

Recall that $M > 0$ is some constant such that $\mathcal{A} = [0,M]^{d^2}$, where $\mathcal{A}$ is the action space for the MFCP. Under this construction, all of the convexity and Lipschitz continuity constraints in Assumptions (A) – (C) are satisfied [2].

For the sake of comparison, we utilize the same architecture as in [1]; a neural network with four hidden LSTM-like layers and one dense output layer, with tanh activation throughout. We performed hyperparameter tuning on the number of layers, the number of samples used in each epoch, the width of the network, and the learning rate schedule. Unless otherwise noted, all results were produced by training the neural network for 200 epochs, with 10 gradient steps in each epoch, and $K = 10000$ samples for each epoch. After hyperparameter tuning, we found that a cosine one-cycle learning rate schedule, with a peak learning rate of $\alpha = 0.0008$, achieved the best performance. All experiments were performed with the Adam optimizer with weight decay and gradient clipping, which we found improved performance.

The plots in Figure 4.1 and Figure 4.2 below demonstrate the value functions approximated by DGM with L^∞-loss and L^2-loss respectively in dimension $d = 2$. Both our DGM algorithm (with the L^∞-loss) and the DGM algorithm with L^2-loss perform similarly, accurately solving the two-dimensional HJB equation. Figures 4.4 and 4.3 contain the corresponding loss curves. We remark that the loss curves for the two methods are displayed on different scales because they are *not* comparable metrics of the performance of the solver. Given the formulation of the L^∞-loss in Equation 3.2, we expect the L^∞-loss to converge to a higher value than the L^2-loss, even if both metrics approximately solve the HJB equation.

As demonstrated in Figures 4.4 and 4.3, both metrics are susceptible to local minima using the LSTM-like architecture outlined in [1]. Given sufficient training time and appropriate hyperparameter tuning, including the number of samples at each step, the learning rate schedule and the optimizer in use, both algorithms can closely approximate the true terminal condition of the example problem as demonstrated in Figure 4.5. We expect that a deeper neural network, coupled with increased training time, would further improve accuracy. However, we defer such studies to a future work, having demonstrated the validity of the algorithm with a relatively simple architecture here.

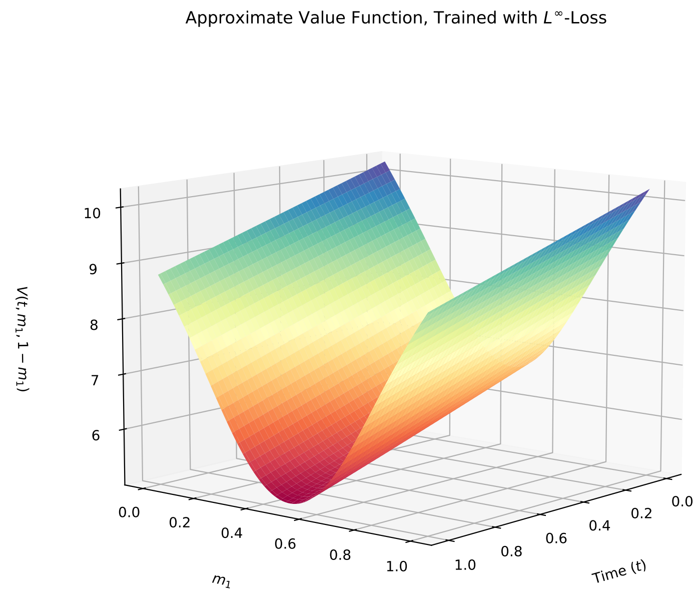

Figure 4.1 Approximate value function, learned using L^∞-loss, in dimension $d = 2$.

In Figure 4.6, we demonstrate the relationship between the sample size K and the stability of the loss for the uniform DGM algorithm. Unless otherwise specified, all numerical tests are carried out with $K = 10000$, as the tradeoff between stability and runtime becomes worse as the number of samples exceeds $K = 10000$. As K increases, we observe more stable training, as expected.

Finally, in Table 4.1, we demonstrate the scalability of the DGM algorithm, implemented in JAX and run with GPU acceleration. By utilizing JAX's built-in auto-differentiation, just-in-time compilation, and GPU acceleration, the DGM algorithm

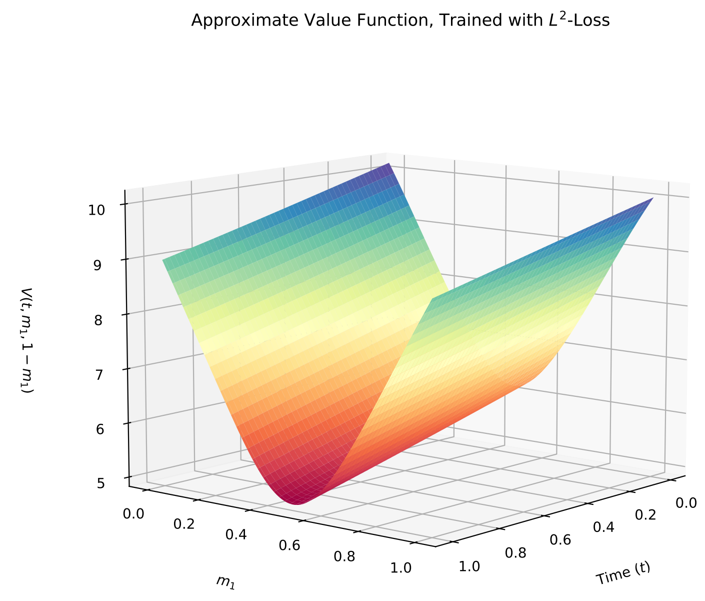

Figure 4.2 Approximate value function, learned using L^2-loss, in dimension $d = 2$.

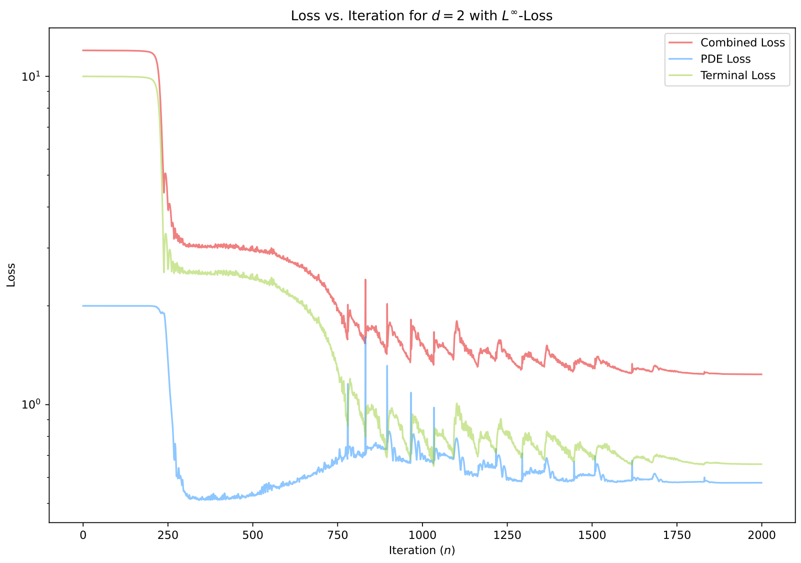

Figure 4.3 L^∞ training loss in dimension $d = 2$. PDE loss refers to the first term in the L^∞-loss from Equation (3.2), terminal loss refers to the second term in Equation (3.2), and the combined loss represents the entirety of Equation (3.2).

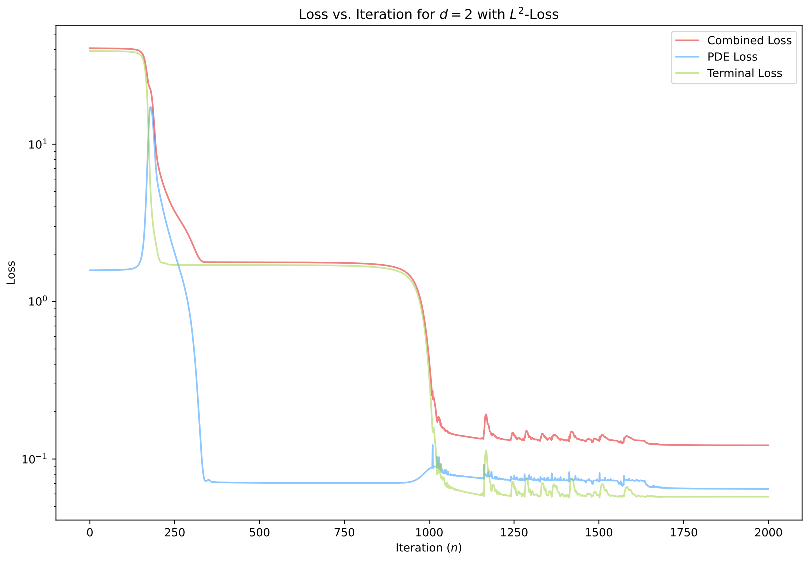

Figure 4.4 L^2 training loss in dimension $d = 2$. PDE loss refers to the first term in the L^2-loss from Equation (3.20), terminal loss refers to the second term in Equation (3.20), and the combined loss represents the entirety of Equation (3.20).

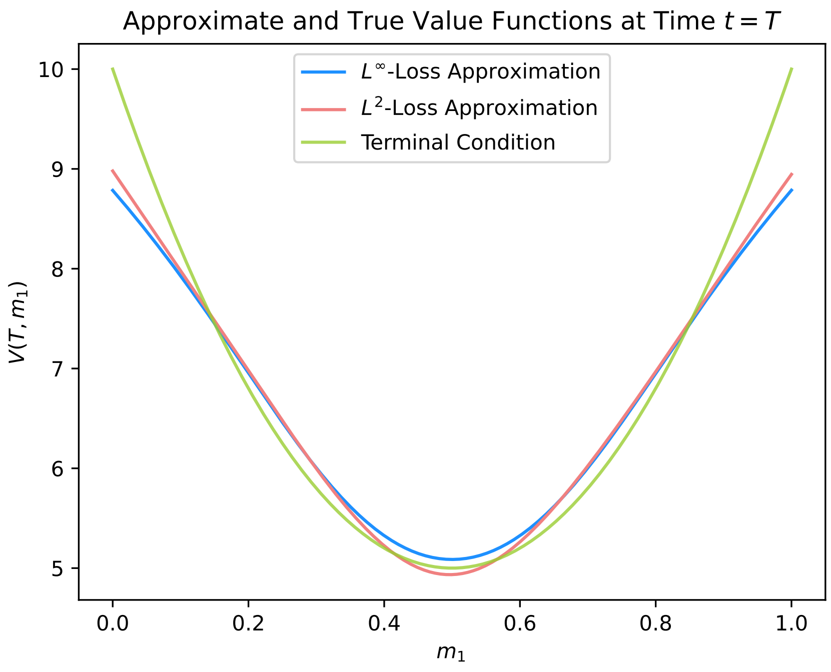

Figure 4.5 Approximate and true terminal conditions for both DGM algorithms in dimension $d = 2$.

scales well to dimension $d = 200$, with roughly linear increases in runtime past dimension $d = 100$. By runtime, we refer to the time that it takes to train the DGM network with uniform loss for 200 epochs, with 10 gradient descent steps in each epoch, and $K = 10000$ for each epoch. As the dimension increases, we observe that the loss decreases, as sampling points near the boundary of the simplex occurs with much lower probability in higher dimensions. Indeed, Figure 4.5 demonstrates that neither the L^∞-loss nor the L^2-loss is able to fully learn the terminal condition near the boundary of the simplex, but this has a smaller impact on the loss in higher dimensions, thus resulting in lower losses as the dimension increases. However, as in dimension $d = 2$, we still expect that the neural network learns the solution to the HJB equation well away from the boundary of the simplex.

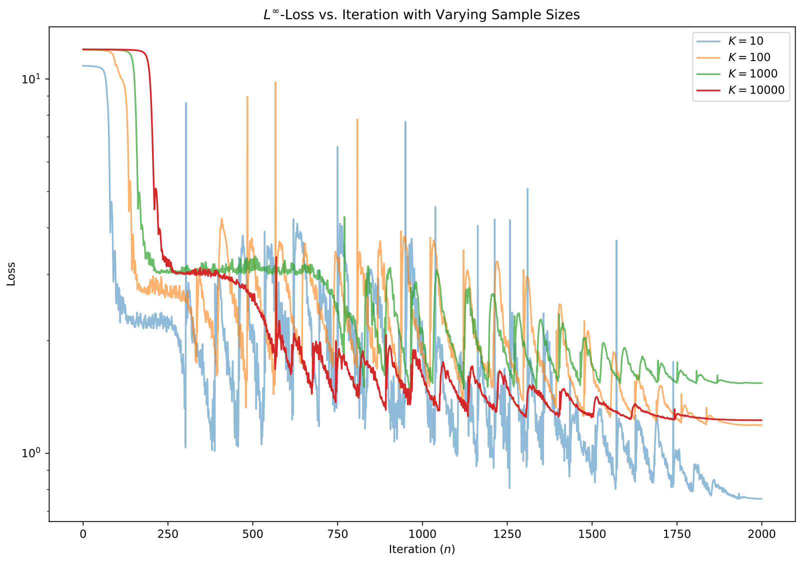

Figure 4.6 Dependence of L^∞-loss on number of samples K. As the number of samples increases, the training process becomes more stable. Note that, with fewer samples, a lower loss does *not* reflect a more accurate solution, as the sampled points are likely do not reflect the validity of the approximation on the entire simplex.

4.2 Improved Sampling Via Adversarial Training

In this section, we briefly return to the error bound presented in Proposition 3.1.1 of Section 3.1. There, we showed that the expected difference between the sampled L^∞-loss and the true L^∞-loss is $\mathcal{O}(K^{-1/(d-1)})$. As noted in Section 3.1, this bound is likely

Table 4.1: Uniform DGM training times and losses as dimension d increases. PDE loss refers to the first term in the DGM loss from Equation (3.2), while terminal loss refers to the second term in Equation (3.2).

Dimension d	Training Time (s)	Combined Loss	PDE Loss	Terminal Loss
2	47.5	1.2134	0.5452	0.6682
5	56.1	0.7417	0.2508	0.4909
10	62.0	0.7200	0.1190	0.6010
20	65.0	0.2772	0.0392	0.2380
50	76.0	0.0588	0.0093	0.0494
100	100.0	0.0217	0.0031	0.0186
200	221.0	0.0070	0.0011	0.0059

pessimistic, as the numerical experiments in Section 4.1 exhibit convergence with approximately $K = 10000$ samples. In practice, it may be possible to more closely approximate the true L^∞-loss and improve the training procedure of the DGM algorithm via adversarial training. Adversarial training, in the context of neural network solutions to high-dimensional PDE, refers to a class of SGD-based algorithms in which, prior to each standard SGD step, an extra gradient step that biases training towards points that maximize the chosen loss functional is performed. A modified DGM algorithm that incorporates adversarial training is presented in Algorithm 2.

Algorithm 2 Uniform DGM with Adversarial Training

```
Initialize parameters $\theta^{(0)} \in \mathbb{R}^P$
Initialize tolerance $\delta \in (0,1)$
$n \leftarrow 0$
Uniformly sample $(t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K} \in [0,T] \times S_d \times S_d$
while $G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta^{(n)}) \geq \delta$ do
    Sample $(t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K} \in [0,T] \times S_d$
    for $j = 1, \ldots, K$ do
        $t^{(j)} \leftarrow \text{Proj}_{[0,T]} \left(t^{(j)} + \eta \text{sign} \nabla_t (\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)}))^2\right)$
        $m^{(j)} \leftarrow \text{Proj}_{S_d} \left(m^{(j)} + \eta \text{sign} \nabla_m (\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)}))^2\right)$
        $p^{(j)} \leftarrow \text{Proj}_{S_d} \left(p^{(j)} + \eta \text{sign} \nabla_p (\varphi(T, p^{(j)}; \theta) - G(p^{(j)}))^2\right)$
    end for
    $\theta^{(n+1)} \leftarrow \theta^{(n)} - \alpha^{(n)} \nabla_\theta G((t^{(j)}, m^{(j)}, p^{(j)})_{j=1,\ldots,K}, \theta^{(n)})$
    $n \leftarrow n+1$
end while
```

Above, $\eta > 0$ is a learning rate hyperparameter for the adversarial training step of the algorithm. Compared to the original DGM algorithm with L^∞ loss, presented in Algorithm 1, adversarial training only requires a slight modification. Namely, before updating the parameters via a gradient step with respect to the L^∞-loss, the above algorithm takes a gradient a point that (locally) maximizes the L^∞-loss. This is reflected by the updates of the form

$$t^{(j)} \leftarrow \mathrm{Proj}_{[0,T]}\left(t^{(j)} + \eta \mathrm{sign} \nabla_t(\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)}))^2\right),$$

for $j = 1, \dots, K$. For instance, the gradient $\nabla_t(\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)})$ is in the direction of a point that (locally) maximizes the PDE loss, with respect to the time variable. Once this update step has been performed, the modified sample is projected back onto the domain of interest. In the case of the time variable, this corresponds to a projection according to the function

$$\mathrm{Proj}_{[0,T]}(t) = \begin{cases} 0 & t \leq 0, \\ t & t \in (0,T), \\ T & t \geq T. \end{cases}$$

Similarly, for the sampled points on the simplex, we perform adversarial updates of the form

$$m^{(j)} \leftarrow \mathrm{Proj}_{S_d}\left(m^{(j)} + \eta \mathrm{sign} \nabla_m(\mathcal{L}[\varphi(\cdot,\cdot;\theta)](t^{(j)}, m^{(j)}))^2\right)$$

and

$$p^{(j)} \leftarrow \mathrm{Proj}_{S_d}\left(p^{(j)} + \eta \mathrm{sign} \nabla_p(\varphi(T,p;\theta) - G(p))^2\right),$$

both for $j = 1, \dots, K$. Exactly as before, both updates return a modified sample in the direction of a point that locally maximizes the PDE loss and the terminal loss respectively. Now, the function Proj_{S_d} is the projection onto the simplex S_d with respect to the Euclidean norm. In general, this projection does not have an analytical expression, but several efficient procedures exist to compute it numerically [60].

Adversarial training is itself a standard heuristic method in machine learning [61] that aims to improve robustness. However, the technique was only recently applied to neural network-based PDE solvers, as in [59, 62, 63]. As in Section 3.4, previous work that applies adversarial training to solve high-dimensional PDE is typically limited to the case of second-order, parabolic HJB equations. Applications of adversarial training to Poisson's equation and the Allen–Cahn equation are also explored in [63], but thus far, no work has studied applications of this method to first-order HJB equations that arise from MFCPs. We remark, however, that this approach is purely heuristic, and the impact that it may have on convergence is currently unknown. Although empirical studies for second-order HJB equations with analytical solutions in [59, 62] illustrate improvements over standard, DGM-type algorithms, there is no work that currently quantifies the impact that the additional gradient step in Algorithm 2 has on the error bound in Proposition 3.1.1. Recent theoretical work in [64] attempts to rigorously characterize the training dynamics of adversarially-trained neural networks, but the results therein remain limited to the case of random deep neural networks. Future work may attempt to describe the training dynamics of adversarial training when applied to DGM-type algorithms.

Chapter 5

Conclusions

We present a novel method for solving high-dimensional HJB equations arising from MFCPs, complete with a convergence proof our algorithm. Our algorithm, which utilizes an L^∞-loss functional for training rather than a standard L^2-loss functional, allows us to leverage the theory of viscosity solutions to prove that if a section of neural network approximators takes the L^∞-loss functional to zero, then the sequence must uniformly converge to the unique viscosity solution of the HJB equation. Notably, our approach does *not* assume boundedness and equicontinuity of the sequence of neural network approximators as in [1], and we thus provide a more general convergence guarantee than previous work that leverages the DGM to solve high-dimensional HJB equations. Our work provides a rigorous foundation for future, deep learning-based research into numerical solutions for the MFCP.

There exists significant potential for future work in this area, both of theoretical and computational nature. For instance, because many HJB equations do not admit classical solutions, relaxing the regularity assumptions presented in Section 2 is of utmost importance. Establishing the convergence of the DGM (or a related algorithm) for HJB equations that only admit viscosity solutions, as outlined in [2], would open a new avenue of research for approximating weak solutions to high-dimensional PDEs using machine learning *and* provide a guarantee that is applicable to a more general class of MFCPs. Similarly, future work may investigate the convergence and performance of deep BSDE and related deep backwards schemes, as introduced in [26]. The convergence of a deep backwards scheme for mean-field games was recently established in [14], and it is likely that a similar approach applies to MFCPs. On the computational side, Section 4.2 presents a possible avenue for future work that aims to improve the efficiency of the

DGM. Importantly, the DGM is a sampling-based algorithm in practice. Consequently, although it is far computationally-tractable than discretization-based PDE solvers in high dimensions, the DGM still suffers from dimensionality-related effects (due to sampling) that may be abated by better training schemes. Investigating convergence properties of adversarial training schemes is also of theoretical and practical interest.

References

[1] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.08.029. URL `https://www.sciencedirect.com/science/article/pii/S0021999118305527`.

[2] Alekos Cecchin. Finite state N-agent and mean field control problems. *ESAIM: COCV*, 27:31, 2021. doi: 10.1051/cocv/2021032. URL `https://doi.org/10.1051/cocv/2021032`.

[3] Min Yang, Guanjun Liu, Ziyuan Zhou, and Jiacun Wang. Partially Observable Mean Field Multi-Agent Reinforcement Learning Based on Graph Attention Network for UAV Swarms. *Drones*, 7(7):476, 2023. ISSN 2504-446X. doi: 10.3390/drones7070476. URL `https://www.mdpi.com/2504-446X/7/7/476`.

[4] Médéric Motte and Huyên Pham. Mean-field Markov decision processes with common noise and open-loop controls. *Ann. Appl. Probab.*, 32(2):1421–1458, 2022. ISSN 1050-5164. doi: 10.1214/21-aap1713. URL `https://doi-org.proxy.lib.umich.edu/10.1214/21-aap1713`.

[5] Jian Li, Rajarshi Bhattacharyya, Suman Paul, Srinivas Shakkottai, and Vijay Subramanian. Incentivizing sharing in realtime D2D streaming networks: A mean field game perspective. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2119–2127, 2015. doi: 10.1109/INFOCOM.2015.7218597.

[6] Martin Burger, Marco Di Francesco, Peter A. Markowich, and Marie-Therese Wolfram. On a mean field game optimal control approach modeling fast exit scenarios in human crowds. In *52nd IEEE Conference on Decision and Control*, pages 3128–3133, 2013. doi: 10.1109/CDC.2013.6760360.

[7] Christian Dogbé. Modeling crowd dynamics by the mean-field limit approach. *Mathematical and Computer Modelling*, 52(9–10):1506–1520, November 2010. ISSN 0895-7177. doi: 10.1016/j.mcm.2010.06.012. URL `http://dx.doi.org/10.1016/j.mcm.2010.06.012`.

[8] Ken R Duffy. Mean field Markov models of wireless local area networks. *Markov Processes and Related Fields*, 16(2):295–328, 2010.

[9] Vivek S. Borkar and Rajesh Sundaresan. Asymptotics of the invariant measure in mean field models with jumps. *Stoch. Syst.*, 2(2):322–380, 2012. doi: 10.1214/12-SSY064. URL `https://doi.org/10.1214/12-SSY064`.

[10] Robert J. Aumann. Markets with a continuum of traders. *Econometrica*, 32:39–50, 1964. ISSN 0012-9682.

[11] David Schmeidler. Equilibrium points of nonatomic games. *J. Statist. Phys.*, 7: 295–300, 1973. ISSN 0022-4715.

[12] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260, 2007. doi: 10.1007/s11537-007-0657-8. URL `https://doi.org/10.1007/s11537-007-0657-8`.

[13] Minyi Huang, Roland P. Malhamé, and Peter E. Caines. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221 – 252, 2006.

[14] Asaf Cohen, Mathieu Laurière, and Ethan Zell. Deep Backward and Galerkin Methods for the Finite State Master Equation, 2024.

[15] René Carmona and François Delarue. Probabilistic Analysis of Mean-Field Games. *SIAM Journal on Control and Optimization*, 51(4):2705–2734, 2013. doi: 10.1137/120883499. URL `https://doi.org/10.1137/120883499`.

[16] Daniel Andersson and Boualem Djehiche. A Maximum Principle for SDEs of Mean-Field Type. *Applied Mathematics & Optimization*, 63(3):341–356, 2011. doi: 10.1007/s00245-010-9123-8. URL `https://doi.org/10.1007/s00245-010-9123-8`.

[17] Huyên Pham and Xiaoli Wei. Dynamic Programming for Optimal Control of Stochastic McKean–Vlasov Dynamics. *SIAM Journal on Control and Optimization*, 55 (2):1069–1101, January 2017. ISSN 1095-7138. doi: 10.1137/16m1071390. URL http://dx.doi.org/10.1137/16M1071390.

[18] Huyên Pham and Xiaoli Wei. Bellman equation and viscosity solutions for mean-field stochastic control problem. *ESAIM: Control, Optimisation and Calculus of Variations*, 24(1):437–461, January 2018. ISSN 1262-3377. doi: 10.1051/cocv/2017019. URL http://dx.doi.org/10.1051/cocv/2017019.

[19] Alain Bensoussan, Jens Frehse, and Phillip Yam. *Mean Field Games and Mean Field Type Control Theory*. Springer New York, 2013. ISBN 9781461485087. doi: 10.1007/978-1-4614-8508-7. URL http://dx.doi.org/10.1007/978-1-4614-8508-7.

[20] René Carmona and François Delarue. *Probabilistic Theory of Mean Field Games with Applications I*. Springer International Publishing, 2018. ISBN 9783319589206. doi: 10.1007/978-3-319-58920-6. URL http://dx.doi.org/10.1007/978-3-319-58920-6.

[21] René Carmona and François Delarue. *Probabilistic Theory of Mean Field Games with Applications II*. Springer International Publishing, 2018. ISBN 9783319564364. doi: 10.1007/978-3-319-56436-4. URL http://dx.doi.org/10.1007/978-3-319-56436-4.

[22] Lars Ruthotto, Stanley J. Osher, Wuchen Li, Levon Nurbekyan, and Samy Wu Fung. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences*, 117(17):9183–9193, April 2020. ISSN 1091-6490. doi: 10.1073/pnas.1922204117. URL http://dx.doi.org/10.1073/pnas.1922204117.

[23] Jean-Pierre Fouque and Zhaoyu Zhang. Deep Learning Methods for Mean Field Control Problems With Delay. *Frontiers in Applied Mathematics and Statistics*, 6:11, 2020. ISSN 2297-4687. doi: 10.3389/fams.2020.00011. URL https://www.frontiersin.org/articles/10.3389/fams.2020.00011.

[24] Diogo A. Gomes, Joana Mohr, and Rafael Rigão Souza. Continuous Time Finite State Mean Field Games. *Applied Mathematics & Optimization*, 68(1):99–143, 2013.

[25] Vassili N. Kolokoltsov. Nonlinear Markov Games on a Finite State Space (Mean-field and Binary Interactions). *International Journal of Statistics and Probability*, 1(1): 77–91, Apr 2012. doi: 10.5539/ijsp.v1n1p77. URL `http://dx.doi.org/10.5539/ijsp.v1n1p77`.

[26] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.*, 5(4):349–380, 2017. ISSN 2194-6701,2194-671X. doi: 10.1007/s40304-017-0117-6. URL `https://doi.org/10.1007/s40304-017-0117-6`.

[27] Huyên Pham. Feynman-Kac representation of fully nonlinear PDEs and applications. *Acta Math. Vietnam.*, 40(2):255–269, 2015. ISSN 0251-4184,2315-4144. doi: 10.1007/s40306-015-0128-x. URL `https://doi.org/10.1007/s40306-015-0128-x`.

[28] Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear PDEs. *Math. Comp.*, 89(324):1547–1579, 2020. ISSN 0025-5718,1088-6842. doi: 10.1090/mcom/3514. URL `https://doi.org/10.1090/mcom/3514`.

[29] René Carmona and Mathieu Laurière. Deep Learning for Mean Field Games and Mean Field Control with Applications to Finance, 2021. URL `https://arxiv.org/abs/2107.04568`.

[30] Mathieu Laurière, Sarah Perrin, Sertan Girgin, Paul Muller, Ayush Jain, Theophile Cabannes, Georgios Piliouras, Julien Pérolat, Romuald Élie, Olivier Pietquin, and Matthieu Geist. Scalable Deep Reinforcement Learning Algorithms for Mean Field Games, 2022. URL `https://arxiv.org/abs/2203.11973`.

[31] Mo Zhou, Jiequn Han, and Jianfeng Lu. Actor-Critic Method for High Dimensional Static Hamilton–Jacobi–Bellman Partial Differential Equations based on Neural Networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066, January 2021. ISSN 1095-7197. doi: 10.1137/21m1402303. URL `http://dx.doi.org/10.1137/21M1402303`.

[32] M. G. Crandall, L. C. Evans, and P. L. Lions. Some properties of viscosity solutions of

Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282 (2):487–502, 1984. ISSN 00029947. URL http://www.jstor.org/stable/1999247.

[33] Michael G. Crandall, Hitoshi Ishii, and Pierre-Louis Lions. User's guide to viscosity solutions of second order partial differential equations. *Bull. Amer. Math. Soc. (N.S.)*, 27(1):1–67, 1992. ISSN 0273-0979,1088-9485. doi: 10.1090/S0273-0979-1992-00266-5. URL https://doi.org/10.1090/S0273-0979-1992-00266-5.

[34] Huyên Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, Berlin, 2009. ISBN 978-3-540-89499-5. doi: 10.1007/978-3-540-89500-8. URL https://doi.org/10.1007/978-3-540-89500-8.

[35] Olav Kallenberg. *Foundations of modern probability*. Probability and its Applications (New York). Springer-Verlag, New York, second edition, 2002. ISBN 0-387-95313-2. doi: 10.1007/978-1-4757-4015-8. URL http://dx.doi.org/10.1007/978-1-4757-4015-8.

[36] W.H. Fleming and H.M. Soner. *Controlled Markov Processes and Viscosity Solutions*. Stochastic Modelling and Applied Probability. Springer New York, 2006. ISBN 9780387310718. URL https://books.google.com/books?id=4Bjz2iWmLyQC.

[37] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

[38] Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the Maximal Loss: How and Why. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 793–801, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/shalev-shwartzb16.html.

[39] K. Biswas, S. Kumar, S. Banerjee, and A. Pandey. Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,

pages 784–793, Los Alamitos, CA, USA, Jun 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.00087. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00087.

[40] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[41] Iosif Pinelis. Order statistics on the spacings between order statistics for the uniform distribution, 2019.

[42] A. Erdélyi and F. G. Tricomi. The asymptotic expansion of a ratio of gamma functions. *Pacific Journal of Mathematics*, 1(1):133 – 142, 1951.

[43] Franco Scarselli and Ah Chung Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(97)00097-X. URL https://www.sciencedirect.com/science/article/pii/S089360809700097X.

[44] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[45] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2(4):303–314, 1989. ISSN 0932-4194,1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

[46] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(90)90005-6. URL https://www.sciencedirect.com/science/article/pii/0893608090900056.

[47] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL `https://www.sciencedirect.com/science/article/pii/089360809190009T`.

[48] M. Stinchcombe and H. White. Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 7–16 vol.3, 1990. doi: 10.1109/IJCNN.1990.137817.

[49] H. N. Mhaskar and Charles A. Micchelli. Degree of approximation by neural and translation networks with a single hidden layer. *Adv. in Appl. Math.*, 16(2):151–183, 1995. ISSN 0196-8858,1090-2074. doi: 10.1006/aama.1995.1008. URL `https://doi.org/10.1006/aama.1995.1008`.

[50] Tim De Ryck, Samuel Lanthaler, and Siddhartha Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2021.08.015. URL `https://www.sciencedirect.com/science/article/pii/S0893608021003208`.

[51] Denis Belomestny, Alexey Naumov, Nikita Puchkin, and Sergey Samsonov. Simultaneous approximation of a smooth function and its derivatives by deep neural networks with piecewise-polynomial activations. *Neural Networks*, 161:242–253, 2023. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2023.01.035. URL `https://www.sciencedirect.com/science/article/pii/S0893608023000473`.

[52] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Mach. Learn.*, 110(2):393–416, feb 2021. ISSN 0885-6125. doi: 10.1007/s10994-020-05929-w. URL `https://doi.org/10.1007/s10994-020-05929-w`.

[53] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32.

Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/95e1533eb1b20a97777749fb94fdb944-Paper.pdf.

[54] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. Training Robust Neural Networks Using Lipschitz Bounds. *IEEE Control Systems Letters*, 6:121–126, 2022. doi: 10.1109/LCSYS.2021.3050444.

[55] Lawrence C. Evans. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, second edition, 2010. ISBN 978-0-8218-4974-3. doi: 10.1090/gsm/019. URL https://doi.org/10.1090/gsm/019.

[56] Maria Michaela Porzio. Existence of solutions for some "noncoercive" parabolic equations. *Discrete and Continuous Dynamical Systems*, 5(3):553 – 568, 1999. doi: 10.3934/dcds.1999.5.553. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0033416895&doi=10.3934%2fdcds.1999.5.553&partnerID=40&md5=95eaebb2e827c32cae4dc66e3dea5434.

[57] Martina Magliocca. Existence results for a Cauchy–Dirichlet parabolic problem with a repulsive gradient term. *Nonlinear Analysis*, 166:102–143, 2018. ISSN 0362-546X. doi: https://doi.org/10.1016/j.na.2017.09.012. URL https://www.sciencedirect.com/science/article/pii/S0362546X17302390.

[58] P. Chandra and Y. Singh. Feedforward sigmoidal networks – equicontinuity and fault-tolerance properties. *IEEE Transactions on Neural Networks*, 15(6):1350–1366, 2004. doi: 10.1109/TNN.2004.831198.

[59] Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is L^2 Physics Informed Loss Always Suitable for Training Physics Informed Neural Network? In *Advances in Neural Information Processing Systems*, volume 35, pages 8278–8290. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/374050dc3f211267bd6bf0ea24eae184-Paper-Conference.pdf.

[60] Laurent Condat. Fast projection onto the simplex and the ℓ_1-ball. *Mathematical Programming*, 158(1):575–585, 2016. doi: 10.1007/s10107-015-0946-6. URL https://doi.org/10.1007/s10107-015-0946-6.

[61] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent Advances in Adversarial Training for Adversarial Robustness. *CoRR*, abs/2102.01356, 2021. URL `https://arxiv.org/abs/2102.01356`.

[62] Pongpisit Thanasutives, Masayuki Numao, and Ken-ichi Fukui. Adversarial Multi-task Learning Enhanced Physics-informed Neural Networks for Solving Partial Differential Equations. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2021. doi: 10.1109/IJCNN52387.2021.9533606.

[63] Yao Li, Shengzhu Shi, Zhichang Guo, and Boying Wu. Adversarial Training for Physics-Informed Neural Networks, 2023.

[64] Soichiro Kumano, Hiroshi Kera, and Toshihiko Yamasaki. Adversarial Training from Mean Field Perspective. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 75097–75150. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/edcd1aa172dceda2ea9d45a48f25d3e3-Paper-Conference.pdf`.

[65] Philipp Grohs and Lukas Herrmann. Deep neural network approximation for high-dimensional parabolic Hamilton-Jacobi-Bellman equations, 2021.

[66] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(90)90005-6. URL `https://www.sciencedirect.com/science/article/pii/0893608090900056`.

[67] Hitoshi Ishii. Viscosity solutions to nonlinear partial differential equations. *Sūgaku*, 46(2):144–157, 1994. ISSN 0039-470X,1883-6127.

[68] Walter Rudin. *Real and complex analysis.* McGraw-Hill Book Co., New York, third edition, 1987. ISBN 0-07-054234-1.

[69] Lawrence C. Evans. The perturbed test function method for viscosity solutions of nonlinear PDE. *Proc. Roy. Soc. Edinburgh Sect. A*, 111(3-4):359–375, 1989. ISSN

0308-2105,1473-7124. doi: 10.1017/S0308210500018631. URL https://doi.org/10.1017/S0308210500018631.

[70] Daniel Lacker. Limit theory for controlled McKean-Vlasov dynamics. *SIAM J. Control Optim.*, 55(3):1641–1672, 2017. ISSN 0363-0129,1095-7138. doi: 10.1137/16M1095895. URL https://doi.org/10.1137/16M1095895.

[71] René Carmona and François Delarue. Forward-backward stochastic differential equations and controlled McKean-Vlasov dynamics. *Ann. Probab.*, 43(5):2647–2700, 2015. ISSN 0091-1798,2168-894X. doi: 10.1214/14-AOP946. URL https://doi.org/10.1214/14-AOP946.

[72] Jian Li, Jing Yue, Wen Zhang, and Wansuo Duan. The deep learning Galerkin method for the general Stokes equations. *J. Sci. Comput.*, 93(1):Paper No. 5, 20, 2022. ISSN 0885-7474,1573-7691. doi: 10.1007/s10915-022-01930-8. URL https://doi.org/10.1007/s10915-022-01930-8.

[73] Lawrence C. Evans and Ronald F. Gariepy. *Measure theory and fine properties of functions*. Textbooks in Mathematics. CRC Press, Boca Raton, FL, revised edition, 2015. ISBN 978-1-4822-4238-6.

[74] Juha Heinonen. *Lectures on Analysis on Metric Spaces*. Springer New York, 2001. doi: 10.1007/978-1-4613-0131-8. URL https://doi.org/10.1007/978-1-4613-0131-8.

[75] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J. Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110(2): 393–416, December 2020. ISSN 1573-0565. doi: 10.1007/s10994-020-05929-w. URL http://dx.doi.org/10.1007/s10994-020-05929-w.